Бектенова Д.Б. Кыштобаева Т.Т.
Молдошев Р.А. Асанова М.Б.
Мокешов Ж.К.

ПРОГРАММАЛООНУН НЕГИЗДЕРИ

Алгоритмдер
Turbo Pascal
Qbasic

Кыргыз орто мектептери үчүн окуу куралы

Бишкек-2003

УДК 681.3.06(075)

Рецензия бергендер:
Кыргыз Агрардык Университетинин информатика кафедрасынын доценти, физика-математика илимдеринин кандидаты Осмонов К. Т.

Кыргыз мамлекеттик улуттук университетиндеги информациялык технологиялар борборунун директорунун орун басары, доцент Эралиев Ж. Т.

Улуттук компьютердик гимназиянын мугалимдери:
Бектенова Д. Б., Кыштобаева Т. Т., Молдошев Р. А, Асанова М. Б, Мокешов Ж. К.

Программалоонун негиздери. Мектеп окуучулары үчүн окуу куралы.. Б. 2003ж.-205б.

Бул окуу куралы төрт бөлүмдөн турат. I бөлүмдө информатика илиминин кычкача тарыхы, эсептөө системалары, алгоритм жөнүндө кеңири маалымат берилет. II бөлүмдө Турбо Паскаль программалоо системасы, алфавити, структурасы, операторлору, функциялары, процедуралары толугу менен берилет. Ошону менен бирге үн, графика жана файлдар менен иштөө мүмкүнчүлүктөрү да кеңири каралган. III бөлүмдө Qbasic программалоо системасы, структуралык түзүлүшү боюнча II бөлүмдөгүдөй толугу менен каралган. Окуу куралынын IV бөлүмүндө мектеп мугалимдерине өтө зарыл болгон практикалык жана лабораториялык иштерден мисалдар берилген. Ар бир теманын башталышын кыскача теориялык материалдар толуктап турат.

УДК 681.3.06(075)
Бектенова Д. Б., Кыштобаева Т. Т., Молдошев Р. А, Асанова М. Б, Мокешов Ж. К

КИРИШҮҮ

Бүгүнкү күндө компьютерди колдонуу мүмкүнчүлүктөрү өтө кеңири кулач жайды. Компьютердик технологиянын өнүгүшү менен, аны өздөштүрүү талабы жогорулады. Ушуга ылайык информациялык технологияны(компьютерди) ар бир инсан колдонуучу катары билүүгө тийиш. Ал эми компьютердик сабаттуулукту жоюу процесси мектептен башталаары айкын. Ошондуктан компьютерлештирүү жүгүнүн басымдуу бөлүгү орто мектептердин информатика мугалимдерине таандык болуп эсептелет.

Азыркы учурда мектептерибиз компьютерлер менен толук камсыз болбосо да, ошондой эле алардагы компьютерлердин маркасы бирдей болбосо дагы, информатиканын негиздери окшош. Демек информатика сабагын окутууда мугалимдин жеткиликтүү аракеттери сөзсүз керек. Компьютердик сабаттын алгачкы кадамдарын эски маркадагы компьютерден баштаган ар бир окуучу, өз ишин жакшы билген устат мугалиминин жардамы менен Internet дүйнөсүнө эркин саякат жасай алат деп ишенсек болот. Ошондуктан мугалим жана окуучулар үчүн компьютердик сабаттуулукту жоюу максатында информатика сабагы боюнча кыргыз тилиндеги окуу китептеринин болушу азыркы учурга абдан керек экендиги бардыгыбызга белгилүү. Кыргыз тилинде жазылган көптөгөн агай-эжейлерибиздин китепте-ринин саны дагы көбөйүп, дэңгээлдери жогорулай бериши керек экен-дигин турмуш өзү далилдеп отурат. Бул китепти жазуу менен муга-лимдерге жана мектеп окуучуларына жардам берүүгө аракет кылдык.

Мектеп курсунда информатиканы окутууда, б.а. компьютердик сабаттуулукту жоюда компьютерде маселе чыгаруу мүмкүнчүлүктөрү (алгоритмдер, программалоо тилдери) негизги орундарды ээлейт. Ушуларды эске алуу менен китептин структурасында төмөнкү бөлүмдөр каралды:

I бөлүм – Алгоритмдер. Мында эсептөө системалары жана алгоритм тили жөнүндө маалыматтар берилген.

II бөлүм – Turbo Pascal программалоо тили.

III бөлүм – Qbasic программалоо тили.

Акыркы эки бөлүм программалоо тилдерине арналып, коюлган маселелерди компьютерде иштетүү ыкмалары каралды.

Колдонуучулар өз маселелерин компьютерде чыгара алышат. Маселени чыгарып, анын жыйынтыгын көрүү үчүн кандайдыр бир программалоо тилинде анын программасын түзүп, иштетүү керек. Кадимки тилдер сыяктуу эле программалоо тилдеринин көптөгөн түрлөрү(1000ден ашык) бар. Программалоо тили ага байланыштуу көптөгөн сөздөрдөн түзүлүп, адам менен компьютердин ортосундагы байланыш каражаты болуп алат. Программалоо тилдери тилдин символдорунан таяныш, синтаксистик эрежелердин негизинде сөздөрү(лексикалык бирдик), сүйлөмдөрү (операто-

Программалоонун синтаксистик эрежелери, сүйлөмдөр (операторлор) программа түзүүгө бир тарамтуу жол көрсөтөт. Бул чектөөлөр транслятор деп аталуучу атайы программаларга тиешелүү. Транслятор программалык тилде түзүлгөн программаларды машиналык тилге которуучу атайы программа. Тил менен тилдин транслятору система деп айтылат. Системанын компиляциялоочу жана интерпретациялоочу типтери бар. Компилятор программаны толугу менен которуп алып андан кийин аткарат. Ошондой эле эстин аз бөлүгүн ээлеп, тез иштейт. Анда негизинен көптөгөн эсептөөлөрдү аткаруучу илимий техникалык маселелерди атайы даярдалган адистер иштешет. Паскаль, Си, Фортран сыяктуу компилятордук системалар белгилүү.

Интерпретатор командаларды анализдеп, тиешелүү аракеттерди маек түрүндө аткаруучу программа. Программа маек түрүндө аткарылгандыктан ал толук аткарылып бүткөнчө эле туура же туура эмес түзүлгөндүгүн билүүгө болот. Интерпретаторлор эстин көп бөлүгүн ээлеп жай иштейт.

IV бөлүм – Практикалык жана лабораториялык иштер. Бул бөлүм жогоруда каралган түшүнүктөрдү бекемдөө максатында көрсөтүлдү. Ар бир лабораториялык иште каралган материалдар боюнча бирдей мисалдын алгоритми, программасы(Turbo Pascalда, Qbasicте) түзүлүп, тапшырмалар берилди.

Ал эми "Тиркемелерде" негизги керектүү деп эсептелген кошумча түшүнүктөр чагылдырылды.

Бул китеп сиздин информатикага болгон алгачкы кадам таштооңузга чоң жардам болот деген ишеничтебиз. Ийгилик каалайбыз!

I БӨЛҮМ
АЛГОРИТМДЕР

1. ЭСЕПТӨӨ СИСТЕМАЛАРЫ

ЭСЕПТӨӨ ТЕХНИКАСЫНЫН ТАРЫХЫ

XVII кылымда эсептөө аппараты катары – арифмометр түзүлгөн. Баштапкы сандар – мында дөңгөлөктөрдү буруу менен берилүүчү, ручкасын айлантканда ар кандай шестерналар жана валиктер кыймылга келүүчү. Натыйжада цифралары бар атайын дөңгөлөктөр арифметикалык амалдардын бирин аткаруунун натыйжасына туура келүүчү абалга келип калчу. Бул операция болуп "+" кошуу жана "–" кемитүү амалдары эсептелинген. Арифмометр кошуу жана кемитүү амалдарын, ал эми мыкты арифмометрлер көбөйтүү, бөлүүнү да аткара алган.

Арифмометрлерди ойлоп табуучулардын арасында кеңири илимий кызыгууларга ээ, кылымдын атактуу, көрүнүктүү окумуштуулары Б.Паскаль жана Г.Лейбниц болгон. Г. Лейбниц цифралардын тобу менен гана иштеген машиналар жөнүндө эмес, формулаларды, текстөрдү билдирүүчү символдордун тобу менен иштей турган машиналар жөнүндө да жазган. Бул машиналар Г.Лейбницке арифмометрлер арифметикалык амалдарды туура аткаргандай эле, логикалык мүнөздөгү амалдарды да туура аткарууга жөндөмдүү болуп элестелген. Бул идея Г. Лейбництин замандаштарынын көпчүлүгүнө абсурд болуп көрүнгөн. Кийинчерээк жаңылануу үчүн таблица түзүү менен алек болгон англиялык математик Ч. Беббидж азыркы учурдагы эсептөө техникасы үчүн эбегейсиз маанилүү болуучу принципти – эсептөө машинасынын иштөөсүнүн программалык башкаруу принцибинин негизинде жаткан эсептөө машинасынын (өзү аналитикалык машина деп атаган) проектин иштеп чыккан. Ч. Беббидждин новатордук ою анын окуучусу акын Дж. Байрондун кызы Ада Лавлейс тарабынан колго алынып, аныкталган. Бирок Ч. Беббидждин идеясынын практикалык ишке ашырылышы мүмкүн эмес болгон. Анткени бул идея өзүнүн кылымынын техникалык мүмкүнчүлүктөрүнөн табигый түрдө алдыда болгон. Мисалы, корабль куруу эсептөөлөрүндө контогон бюролордун айлар, жылдар бою иштөөсү талап кылынган.

40-жылдарында эсептөө техникасында түп тамырынан болгон өзгөрүүлөр болду. Дөңгөлөк, валик ж.б. электрондук элементтер колдонулган эсептөө машиналары пайда болгон. Негизги элементтери радиолампалар болгон. Электромеханикалык элементтерден электрондуктарга өтүү эсептөө техникасынын тез иштөөсүн жүздөгөн эсе жогорулатты. 1943-жылы американец Эйкен Говард иштеп чыккан 1-электрондук тез иштөөчү эсептөө машинасы болуп Марк-1 эсептелген. ЭЭМ ди иштетүүгө даярдоо учурунда өткөргүчтөрдү туташтыруу үчүн бир нече саат же күндөр керек болгон, ал эми эсеп-төө бир нече минута же секунда ичинде жүргүзүлгөн. Программанын аткарылыш процессин жөнөкөйлөтүү үчүн, программаны өз эсине сактай ала турган компьютердин конструкциясын Моучли жана Дж. Эккерт түзүшкөн. Бул компьютер жөнүндө белгилүү математик Джон фон Нейман 1945-жылы доклад даярдап чыккан. ЭЭМдин кызма-тынын жалпы прицибин формулировкалаган. Докладды түшүнүктүү жана жөнөкөй формулировкалап чыгуу менен жалпыга таанымал болгон. Ошол учурдан баштап көпчүлүк компьютерлер бул принцип менен түзүлө баштаган. 1945-жылдардын аягында АКШда Дж. Эккерт жана Дж. Моучли тарабынан ЭНИАК компьютери түзүлгөн. ЭНИАК "Марк-1" ге караганда 1000 эсе тез иштеген. Бул машинанын салмагы 30 т, аянты 50 $м^2$, баасы 2,8 млн.доллар болгон. Джон фон Неймандын принциби менен 1949-жылы англис изилдөөчүсү Морис Уилксон тарабынан жаңы ЭЭМ түзүлгөн. Мурунку СССРде 1950-жылы академик С.А.Лебедевдин жетектөөсүнүн астында МЭСМ эсептөө машинасы (кичипе электрондук эсептөө машина), 1952-жылы БЭСМ (Чоң электрондук эсептөө машинасы) жасалган. Андан соң электрондук эсептөө машиналарын көптөгөн мамлекеттерде жасаша башташты. Электрондук машиналардын конструкциясы үзгүлтүксүз түрдө жакшырып жатты. 60-жылдары радиолампалар транзисторлор менен алмашты, алардын ордун кийинчерээк кристаллдардын бетине орнотулган микросхемалар басты. Эсептөө машиналары өлчөмү боюнча кичирейди, арзандады, тез иштөөчү, секундасына бир нече миллион операцияга чейин остү. Бул болсо, ал машиналарды илимде, конструктирлөөдө, пландоодо, башкарууда, сурап билүү кызматында, окутуу тармагында кеңири колдонууга мүмкүнчүлүк берди. Бул жерде адамдын илимий жана практикалык ишмердүүлүгүнүн көптөгөн тармактарын санап чыкса болот.

Кийинки мезгилде эсептөө машиналары бүткүл дүйнөдө компьютер деп аталып калды (computer – эсептегич деген англис сөзүнөн). Биздин өлкөдө да бул термин кабыл алынган, бирок мурдагы ЭЭМ(электрондук эсептөө машина) аты деле кездеше берет. Компьютер аталышына караганда эсептөөгө гана багыталган эмес, чындыгында азыркы компьютердин мүмкүнчүлүктөрү алда канча кеңири, алар: графикалык иштерди, тексттерди өзгөртүп түзүү жана

көптөгөн башка мүмкүнчүлүктөр. Бул Ч. Лейбниц 300 жыл мурда айткан идеялардын тууралыгын ырастайт.

ИНФОРМАТИКА ИЛИМИ

Эсептөө техникасынын өнүгүшү жана кеңири таралышы информатика деп аталган илимдин жаңы бөлүгүнүн пайда болушуна түрткү болду.

Информатиканын негизги багыттары татаал изилдөө жана практикалык маселелерди чечүүнүн атайын компьютердик методдорун иштеп чыгуу менен байланышкан. Мурдагы СССР мамлекетинде ЭЭМ дин өнүгүшүнө академиктер В.М. Глушков, А.А.Дородиницын, А.П. Ершов, А.В. Канторович М.В. Келдыш, М.А. Лаврентьев, С.Л. Соболев, А.Н.Тихоновдор чоң салым кошушкан. Кыргызстанда информатика илимине өз салымдарын кошуп жаткан окумуштуулар да жок эмес. Мисалы: А.А Акаев, А.Ж. Жайнаков, П.С. Панков ж.б.

Информатика көптөгөн мезгилдер бою математиканын бөлүгү катары каралып, математиктердин аракети менен өнүгүп келген. Компьютердик методдун спецификациялуулугу информатиканы өзгөчө илим деп айтууга негиз болсо да, бул илим салмактуу түрдө математиканын жетишкендиктерине таянарын белгилей кетүү керек. Анткени табигый жана техникалык илимдер изилдөөчү кубулуштар-ды жана процесстерди математиканын функция, теңдемелер, барабарсыздыктар системасы ж.б. түшүнүктөрү менен айырбаштап алууга мүмкүн жана изилденип жаткан кубулуштар жана процесстер жөнүндө конкреттүү маалымат алыш үчүн, математикалык объектин үстүнөн кандайдыр бир амалдарды жүргүзүү керек.

"Информатика" деген сөз француз тилинен алынган. Information (информация) жана automatigue (автоматика) – информацияны иштетүү дегенди түшүндүрөт. Информатика тез иштетүүчү адамдын илимий техникалык ишмердигинин жаңы области, информациянын касиеттерин, түзүлүшүн, анын пайда болушун, сакталышын, изделишин, өзгөртүлүшүн, жиберилишин, адамдын күндөлүк турмушунда пайдаланышын окута турган илим.

Информация 20-кылымда жаңы түшүнүк катары кирди. Адам менен автоматтын, автомат менен автоматтын, тирүү жана өлүү жаратылыштын, жаныбарлар менен өсүмдүктөр ортосундагы байланыш бар экендигин далилдеди. Информация айлана-чөйрөдөгү объектилер, кубулуштар дегенди билдирет. Ал өз кезегинде бир перс жөнүндө маалымат дегенди түшүндүрөт. Компьютер иш жүзүндө информацияны кайра иштетүү үчүн белгиленген. Сан, таблица, график, текст түрүндөгү кандайдыр бир процесс жөнүндөгү баштапкы информация ушул эле процесс жөнүндө башка информацияга өзгөртүп түзүлүшү мүмкүн. Мисалы, планеталардын өз ара жайланышы жөнүндө

информация, компьютерден бизди кызыктырган убакыттан кийин байкала турган жайланыш жөнүндө информацияга өзгөртүлө алат. Алгоритм жана программа түшүнүктөрүн жалпысынан талдап чыгалы.

ЭСЕПТӨӨ СИСТЕМАЛАРЫ

Сандарды жазуу жүзүндө көрсөтүү мезгили биздин эрага чейинки 3700-3000 жылы башталган. Шумердиктер чополорго шынаа түрүндөгү белгилерди колдонуп жазышкан. Кийин Египеттик, Римдик, Индиялык, Арабдык эсептөө системалары пайда болгон. Бизге кеңири тараган ондук эсептөө системалары биздин заманга чейинки 500-жылы Индияда пайда болгон.

Сандарды жазып үчүн пайдаланылган ар түрдүү белгилердин тобу **эсептөө системалары** деп аталат.

Эсептөө системасынын төмөнкүдөй түрлөрү кездешет:

1. Экилик эсептөө системасы (0, 1);
2. Сегиздик эсептөө системасы (0, 1, 2, 3, 4, 5, 6, 7);
3. Ондук эсептөө системасы(0, 1, 2, 3, 4, 5, 6, 7, 8, 9);
4. Он алтылык эсептөө системасы (0,1,2,3,4,5,6,7,8,9, a, b, c, d, e, f);

Аталган эсентөө системасына керек болгон белгилердин саны ошол эсептөө системасынын негизи деп аталат.

Эсептөө системасынын негизин латын тамгасы q менен белгилеп жүрөбүз. Мисалы: 56_{10}, 101101_2

Эсептөө системасынын 2 түрү бар.

- Позициялык эсептөө системасы.
- Позициялык эмес эсептөө системасы.

Эгерде кандайдыр бир эсептөө системасында бир сан берилсе жана ошол сандагы бир цифранын мааниси ал турган позицияга жараша өзгөрсө, анда мындай системаны **позициялык эсептөө системасы** деп аташат.

Мисалы: 738 жана 387 деген сандарды алсак, 7 – жүздүктү, 3 – ондукту, 8 – бирдикти көрсөтөт. Ал эми экинчи сандагы ушул эле цифралар 3 – жүздүктү, 8 – ондукту, 7 – бирдикти көрсөтөт.

Позициялык эсептоо системасындагы шартты канаттандырбаса, анда мындай эсептоо системасы **позициялык эмес эсептөө системасы** деп аталат.

Мисалы: IV жана VI деген сандары берилсин. 1-санда I – бирди, V – бешти билдирет. Ошондуктан римдин IV деген саны 4 дегенди билдирет. Экинчи санда деле V – бешти, I – бирди билдирет, I жана V тин маанилери ал турган позицияга жараша өзгөрбөдү, бирок VI саны бул учурда 6 дегенди билдирет.

ЭКИЛИК ЭСЕПТӨӨ СИСТЕМАСЫ

Экилик эсептөө системасында 0 жана 1 деген эки цифра гана колдонулат. Экилик эсептөө системасында арифметикалык амалдарды аткаруунун таблицасы:

Кошуу	Кемитүү	Көбөйтүү
0 + 0 = 0	0 - 0 = 0	0 * 0 = 0
0 + 1 = 1	10 - 1 = 1	0 * 1 = 0
1 + 0 = 1	1 - 0 = 1	1 * 0 = 0
1 + 1 = 10	1 - 1 = 0	1 * 1 = 1

Экилик эсептөө системасында 1+1=10 болот, себеби экилик эсептөө системасы эки цифрадан(0,1) гана турат.

Ал эми 0 - 1 арифметикалык амалдарды аткарууга мүмкүн болбогондуктан, алдында турган жогорку разрядагы сандан карыз сурайт, экилик системада ал 2 ге, башкача айтканда 1 жана 0 го барабар. Ошондуктан 10 - 1 = 1 болот.

Экилик эсептөө системасында сандарды кошуу жана кемитүү төмөнкүчө жүргүзүлөт. Мисалы:

1. $1011_{(2)} + 11011_{(2)} = 100110_{(2)}$ 2. $101101_{(2)} - 10011_{(2)} = 11010_{(2)}$

$$\begin{array}{r} 1011_{(2)} \\ +\,11011_{(2)} \\ \hline 100110_{(2)} \end{array} \qquad \begin{array}{r} 101101_{(2)} \\ -\,10011_{(2)} \\ \hline 11010_{(2)} \end{array}$$

Экилик эсептөө системасында сандарды көбөйтүү жана бөлүү төмөнкүчө жүргүзүлөт.Мисалы:

1. $1101_{(2)} * 101_{(2)} = 1000001_{(2)}$ 2. $110111_{(2)} : 1011_{(2)} = 101_{(2)}$

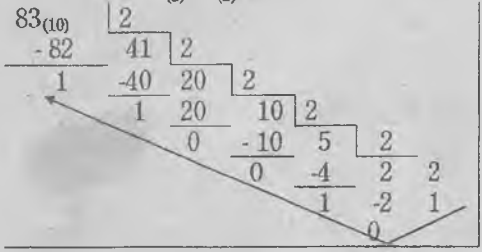
$$\begin{array}{r} 1101_{(2)} \\ *\ \ 101_{(2)} \\ \hline 1101 \\ +\,0000 \\ \hline 1101 \\ \hline 1000001_{(2)} \end{array} \qquad \begin{array}{r|l} 110111_{(2)} & 1011_{(2)} \\ \cline{2-2} -\,1011 & 101_{(2)} \\ \cline{1-1} 1011 & \\ -1011 & \\ \cline{1-1} 0 & \end{array}$$

Ондук эсептөө системасынан экилик эсептөө системасына өтүүнү төмөнкүчө карасак болот.

Ондук эсептөө системасындагы сандан экилик системадагы санга өтүү үчүн, ондук системада берилген санды экиге бөлөбүз. Эгерде пайда болгон тийинди экиден кичине же ага барабар болсо, анда тийиндини бөлүүнү качан экиден кичине болгонго чейин уланта беребиз. Качан тийинди экиден кичине болгон учурда бөлүү процессин токтотуп, тийиндиден баштап жебенин багыты боюнча төмөндөп

жогору көздөй калдыктарды жазуу менен тиешелүү экилик санды алабыз.

Мисалы: $83_{(2)}=X_{(2)}$



Демек, $83_{(2)}=X_{(2)}$ нин мааниси $\mathbf{83_{(10)}=1010011_{(2)}}$

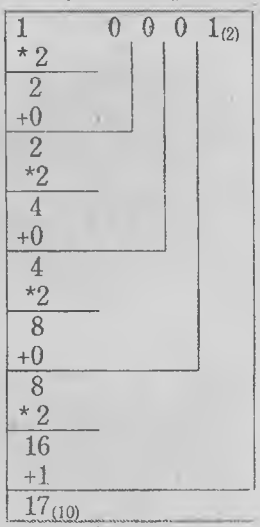
Экилик эсептөө эсептөө системасынан ондук системасына өтүүнү төмөндөгүдөй карайбыз.

Экилик системада берилген сандан ондук системадагы санга өтүү үчүн экилик сандын башкы разрядында турган цифраны экиге көбөйтөбүз, найда болгон көбөйтүндүнү кийинки разряддагы цифрага кошобуз, бул процессти көбөйтүүгө эң акыркы разряд-дагы экилик цифра кошулмайынча уланта беребиз.

Мисалы: $10001_{(2)}=X_{(10)}$

1-жолу:

$10001_{(2)} = 1\bullet2^4 + 0\bullet2^3 + 0\bullet2^2 + 0\bullet2^1 + 1\bullet2^0 = =1\bullet16 +$
$0\bullet8 + 0\bullet4 + 0\bullet2 + 1\bullet1 =$
$= 16 + 0 + 0 + 0 + 1 = 17_{(10)}$



2-жолу

2. АЛГОРИТМ ТИЛИ

МАСЕЛЕНИ КОМПЬЮТЕРДЕ ЧЫГАРУУНУН ЭТАПТАРЫ

Маселенин математикалык коюлушу. Маселелер тиешелүү адистиктин терминдери менен түшүнүктөрүнүн негизинде коюлат. Маселени чыгаруунун биринчи этабында маселенин коюлушун талдоо керек, кайсыл өзгөрмөлөр берилген, кайсыл өзгөрмөлөрүн маселенин жыйынтыгында табуу керек, кандай чектөөлөр тиешелүү (кандай шарттар коюлат).

Андан сырткары кандай теңдемелерди жана логикалык байланыштарды колдонуу керектигин тактап алуу зарыл. Бул процесс математикалык формулага келтирүү же маселенин математикалык моделин түзүү болуп эсептелет. Бул этапта маселенин шартын формалдуу түрдө так аныктап, негизинен математикалык тилде көрсөтүү керек. Эгерде математикалык маселе болсо, анда формалдуу тилдеги этаптуу аткаруунун зарылчылыгы деле жок. Информатикада компьютер эсептөө маселелеринен башка дагы татаал маселелерди чыгарууда көп колдонулат, ошондуктан жогорку этаптарды аткаруу көп күч жана убакытты талап кылат, маселенин баштапкы бери-лишинин курамы жана мүнөздөмөсү аткарылат. Ошону менен бирге бул программаны түзүүнүн негизги бөлүгүн аяктоо болуп эсептелет. Ар түрдүү шарттар жана чектөөлөр математикалык формада жазылат жана маселени чыгарууда кандай жыйынтык алынаары аныкталат. Маанисине карата маселелерди чыгаруудагы белгилүү методдор жөнүндө маалымат берилет.

Алгачкы берилиштер, **результат** жана алардын ортосундагы байланыштар маселенин модели болуп эсептелет. Бул этаптын аткарылышын төмөнкү маселелерден көрсөк болот.

Класста 30 окуучу бар дейли. Алардын ону жакшы, беши эң жакшы окуйт. Класста канча орто окуган окуучу бар?

Математикалык моделди түзүү үчүн бул маселеге төмөнкүдөй белгилөөлөрдү киргизебиз:

КС – класстагы окуучулардын жалпы саны

КЖ – жакшы окуган окуучулардын саны

КЭ – эң жакшы окуган окуучулардын саны

КО – орто окуган окуучулардын саны

Бул арифметикалык туюнтма:

КО = КС – (КЖ + КЭ), (1)

(1) формула биздин маселеде математикалык модел болот.

Чыгаруу методун табуу жана иштетүү. Бул этап формали-зация этабы менен байланыштуу, анын максаты чыгаруу методунун маалыматтын математикалык моделге берүү(өткөрүү). Биринчи методдо алынган бир нече чыгаруу методдорунун ичинен ыңгайлуусун

тандоо керек. Маселелерди чыгаруунун айрым учурунда ошол методдордун бири дагы туура келбейт. Андай болгон учурда формализация этабына кайрылуу керек жана чыгарыла турган маселени кыскартып, кошумча чектөөлөрдү жана шартарды киргизүү керек. Мындан тышкары түзүл-гөн жаңы методдорду иштетүү керек.

Мисалга: (1) формула(математикалык модел) – чыгаруу методун аныктайт: Берилүүчүлөрдү формулага коюу менен бирге чыгарылышын алабыз.

Алгоритмди жазуу. Тандалып алынган методдун негизинде алгоритм түзүлөт. Алгоритмди тандоодо, туура чечимге келүү үчүн бир нече варианттарды анализдеп кароо керек. Бир эле метод өз учурунда бир нече алгоримтдерде колдонулат татаал маселелердин алгоритмин иштетүүдө кадамдап тактоо методун колдонуу керек. Алгоритмдин структурасынын ар бир кадамы жөнөкөй жана түшүнүктүү болушу керек. Бул этап I бөлүктө толук берилет.

Программаны жазуу. Программанын текстин жазуу үчүн программалоо тилин тандоо зарыл. Алгоритм туура түзүлсө, анда програмалоо жеңил болот. Конкреттүү берилиштерден көз каранды болбостон программа универсалдуу болушу керек. Берилиштерде турактуулардан көрө өзгөрмөлөрдү кодонгон ылайыктуу, себеби баштапкы программада турактуулардын баштапкы маанилери өзгөргөн сайын ар бир операторду өзгөртүүгө туура келет. Бул процедура көп убакытты алуу менен каталарды кетирет.

Программа түзүлгөндөн кийин тестирлөө жана жөндөө бул программанын иштешинин тууралыгын текшерип, кетирилген каталарды оңдоо болуп эсептелет. Программаны туралоо программаны иштетүүнүн убактысынын 20-40% тин түзөт. Тууралоо синтаксистик каталарды тактоодон башталат. Программанын текстин жазганда алгоритималык тилдеги кетирилген формалдык каталар жоготулат. Андан кийин тесттин жардамы менен программанын тууралыгы текшерилет да, программа компьютер тарабынан ат карылгандан кийин жыйынтыгы алынат.

АЛГОРИТМ

Күндөлүк турмушта, жумушта, окуу процессинде бир нече маселелерди чыгарууга туура келет. Маселенин мазмуну сөзсүз түрдө математика жана так илимдер менен байланышкан эмес. Көпчүлүк учурда зарыл түрдө чечүүгө туура келген тигил же бул окуялар болот. Түзүлгөн окуянын бир нече чечилүүчү жолдору болушу мүмкүн. Биз, өмүр бою мына ушундай чыгарылыштарды, анын ичинен рационалдуу болгон чыгарылыштарды алууну окуп үйрөнөбүз.

Ар бирибиз эле дүкөнгө бир нерсе сатып алууга барабыз. Эстеп көрүңүз, сиз дүкөнгө жөнөөдөн мурда эмне кыласыз. Ооба, адегенде сиз өзүңүздө канча акча бар экенин (изделүүчү берилиш) жана эмне сатып алуу керектигин (чыгым) ойлойсуз. Андан кийин сиз сатып алуу үчүн кайсы жол менен барууну жана кайсы дүкөндөргө кирүүнү ойлойсуз. Чындыгында, баруунун бир нече жолу болушу мүмкүн. Анда сиз өзүңүзгө керектүү болгон жолдун шартын аныктайсыз.

Мисалы, сиз дүкөнгө киресиз, ал жерде сизге керектүү товар жок. Бул учурда эмне кылуу керек, кайда баруу керек, же керектүү болгон товарды издөөнү токтотосузбу, ойлоносуз.

Мына мурда ойлонуп коюлган жүрүүнүн эрежелерин математикалык терминологиясында колдонсок, анда аны алгоритм деп айтсак болот. Адам анын алгоритм экендигин ойлобой туруп эле, ар кандай аракеттерди жасайт. Мисалы, чоң адам суудан кантип өтүүнүн жолдорун билет жана ал жолдор боюнча суудан өтө алат. Ал эми өзүнүн кичинекей уулу суудан өтүүнүн жолдорун билбейт. Ал үчүн атасы уулуна суудан өтүүнүн төмөнкудөй алгоритмин үйрөтөт:

1. Суунун жээгине келиңиз
2. Жээктеги кайыкка отуруңуз
3. Суунун карама-каршы жээгин көздөй сүзүңүз
4. Кайыктан түшүңүз.
5. Жээкке чыгыңыз.

Бул алгоритмди сиз суудан өтүү үчүн аткарасыз.

Өзүнүн каалагандай жыйынтыгына жетүү үчүн практикалык түрдө, биздин мээбиз дайыма алгоритм түзүү менен алек болот. Биз белгилүү максатка жетүү үчүн ойлонобуз жана анын жолдорун издейбиз. Дагы бир мисалыбызга кайрылалы. Сиз үйдөн чыктыңыз жана ойлонуп түзүлгөн алгоритм боюнча сатып алуу үчүн дүкөндөргө кире баштадыңыз. Демек, сиз алгоритмди турмушта колдонуп жатасыз, аныкталган программа менен иштеп жатасыз.

Алгоритмди практикалык, чындык түрдо аткаруу программа болот. Мисал катары алгоритмге көптөгөн мисалдарды карасак болот: тамак даярдоо (берилгендер – алгачкы продукталар, даярдоонун рецепти – алгоритм, жыйынтык – жаңы тамак), жолдо жүрүү эрежеси (светафордун үч түсү, өтүүнүн эрежеси, жолдон өтүү). Биздин мээбиз бул маселелерди чыгара алат, себеби берилген берилиштер чектелген.

Качан изделүүчү берилиштер көптүк болгон учурду элестетип көрөбүз. Элестетүү кыйын, максатка жетүү үчүн көптөгөн изилдөөлөр жана эсептөөлөр жүргүзүү керек. Изилденүүчү берилиштер математикалык жана физикалык маселелерди иштүүдө миңдеген ар түрдүү вариацияларуу болот, адамдын күчү жетпейт. Бул учурда компьютерге жардамга барабыз, себеби компьютер ошондой чоң эсептөөлөрдү өзүнө алат. Бул учурда компьютер өзү-нүн функциясын адамга эскертет. Сөзсүз түрдө компьютерге жана алгоритмге туура келген, адам тарабынан түзүлгөн программаны киргизүү керек. Мына ушул учурда гана

компьютер программанын үстүндө канча жолу иштөө керек болсо, ошончо жолу иштейт.

Эне баласына жолдон өтүүнүн эрежесин үйрөтүүдө, ага түшүнүктүү болуш үчүн сөз колдонот жана көрсөтүп берет. Мына ушул учурда баланын эсине бир алгоритм түшөт, компьютерге дагы сөзсүз түрдө алгоритм түшүнүктүү болуш үчүн ал түшүнө ала турган программаны киргизүү керек. Алгоритм түшүнүгүнө математикалык так аныктама берүүгө мүмкүн эмес, бирок ошондой болсо да аны оцой эле түшүнсө болот.

Каалагандай маселенин жыйынтыгын алыш үчүн, ошол маселенин берилиштеринин үстүндө кандай амалдарды кайсы иретте жүргүзүш керек экендигин так көрсөтүүчү эрежелердин (көрсөтмөлөрдүн) жыйындысын **алгоритм** деп айтабыз. Алгоритм сөзү Algozithmi деген латын сөзүнөн которулуп алынган. Бул сөз белгилүү Орто Азиялык окумуштуу Мухамбет Бел Муса-Аль-Хорезминин(787-850жж.) ысмына арналып алынган. Анын арифметика жана алгебра боюнча негиздөөсү XII кылымда латын тилинен которулган, бул батыш Европадагы математиканын өрчүшүнө чоң таасир берген.

Алгоритмизация – бул алгоритмди түзүү процесси.

Программа – бул компьютер түшүнгөн тилде алгоритмди жазуу.

Алгоритмди иштеп чыгууда сөзсүз төмөнкү эрежелерди сактоо керек:

1. Адамга түшүнүктүү болгон кыймылдын удаалаштыгы, кадам артынан кадам тургузуу.
2. Изделүүчү берилиштин мүнөзүн аныктоо – скалярдык же матрицалык, сандык же текстик ж.б.
3. Конкреттүү сандарды колдонбой, өзгөрмөлөр менен белгилөөнү колдонууга аракет кылуу керек. (мисалы: 5*8 дегендин ордуна A*B деп жазуу).
4. Компьютерде изделүүчү берилиштерди киргизүү ордун жана компьютерде жыйынтыкты чыгаруучу ордун көрсөтүү.
5. Маселени чыгарууда бардык формулаларды жана кайсы шартта аткарыла тургандыгын көрсөтүү.

АЛГОРИТМДИН КАСИЕТТЕРИ

Алгоритм өзүнүн аныктамасын ачып көрсөтүүчү төмөндөгүдөй касиеттерге ээ:

Дискреттүүлүк. Алгоритмдин аткарылышы жөнөкөй кадамдардын удаалаштыгына болуштурулөт. Ар бир кадам аткаруучу тарабынан кийинки кадамга чейин аткарылышы керек, б.а. кадамда учуроочу бир чондуктун мааниси мурдагы кадамда кандайдыр бир эреженин жардамы менен аныкталган чондуктун мааниси менен аныкталат. Мисалы: Эки санды кошуу

1) Эки санды киргизүү 2) Суммасын эсептөө

3) Жыйынтыгын чыгаруу

Аныктык, тактык. Алгоритм аткаруучуга ылайыкталган жана ага так, түшүнүктүү жана бир маанилүү болушу керек. Ушуну менен бирге алгоритмдин бир кадамында кездешкен бир чоңдук анын калган кадамында да учурай турган болсо, анда кийинки кадамдарда ал чоңдуктун мааниси же түшүнүгү ошол чоңдуктун мурдагы алган мааниси же түшүнүгүнө байланыштуу деп эсептелинет. Алгоритмдин ушул касиетине байланыштуу алгоритмдин аткарылышы механикалык мүнөздө болот жана чыгарыла турган маселе жөнүндө эч кандай көрсөтмөлөрдү же маалыматтарды талап кылбайт.

Мисалы, Кыргыз тилинде жазылган алгоритм кыргыз тилин билбеген адамга түшүнүктүү эмес.

Массалуулук. Түзүлгөн бир эле алгоритмден алгачкы берилиштери менен айырмаланган бир типтеги бир нечелеген маселелерди чечүүгө болот. Мисалы, $ax+b=c$ теңдемесин чыгаруу алгоритми. Бул алгоритмдин аткарылышында a, b, c нын каалаган маанилери үчүн каалаган сызыктуу теңдемени чыгарууга болот.

Жыйынтыктуулук. Алгоритмдин аткарылышы чектүү сандагы амалдарды аткаруудан турат, ошондой эле жыйынтыкты бериши керек. Мисалы: жогоруда келтирилген "Суудан өтүү" алгоритминде жыйынтыгы суудан өтүү болуп эсептелет.

ЧОҢДУКТАР

Чоңдуктар өзгөрмөлүү жана туракту болуп экиге бөлүнөт. Турак-туу деп алгоритмди аткаруу процессинде, алгоритмдин текстинде көрсөтүлгөн маанисин өзгөртпөгөн чоңдукту айтабыз. Өзгөрмөлүү деп, алгоритмди аткаруу процессинде мааниси өзгөрүлгөн чоңдук аталат.

Алгоритмди аткаруу учурунда, убакыттын ар бир учурунда чоңдуктар адатта кандайдыр бир мааниге ээ. Ал учурдагы маани деп аталат. Алгоритмди аткаруу процессинде чоңдуктар конкреттүү маанини албай да калышы мүмкүн. Мындай чоңдуктар аныкталбаган чоңдуктар деп аталат.

Алгоритмди аткаруу процессинде жардамчы чоңдуктар колдонулат. Бул чоңдуктарды арадагы чоңдуктар деп атайбыз. Эгер алгоритмде эки өзгөрмөнүн маанилерин алмаштыруу туура келсе, анда сөзсүз арадагы чоңдук пайдаланылат. Мисалы: C:=A: A:=B: B:=C. Мында C – арадагы чоңдук.

Математиканын, физиканын мектептик курсунда маанилери натуралдык, бүтүн, аныктык сандар болгон сан чоңдуктары көп жолугат. Ал эми алгоритмдерде таблицалар, тизмелер, тексттер, графиктер, геометриялык фигуралар ж.б. сан эмес чоңдуктар кездешет. Математика жана физикада чоңдуктары көп учурда, латын жана грек алфавитинин бир гана тамгасы менен белгиленет. Алгоритм тилинде

чоңдуктардын аттары үчүн каалагандай тамгалар, тамгалардын жана сөздөрдүн тизмектери алгоритмде чоңдуктардын мааниси менен арналышын түшүндүрүүгө пайдаланылат. Чоңдуктар милдеттерине ылайык натуралдык, бүтүн, анык, литердүү ж.б. типте болушат. Өзгөрмөлүү чоңдуктардын типтери кыскача **нат**(натуралдык), **бтн**(бүтүн), **аныk**(аныk), **лит**(литердик) ж.б. сөздөр менен белгиленет.
Мисалы: A=45 – **бтн**(бүтүн), B=3,5 – **аныk**(аныk), C=5 – **нат**(натуралдык),D$="Адис" – **лит**(литердик)

АЛГОРИТМ ТИЛИ

Алгоритмдерди аткаруу жана бир типте так жазуу үчүн колдонулган белгилөөлөрдүн жана эрежелердин системасы **алгоритм тили** деп аталат. Алгоритм тилинин эрежелери компьютердин программалоо тилдеринин негизинде жатат. Алгоритм тилин үйрөнүп алуу, келечекте программалоо тилдеринин арасынан каалаганын билип алууга жардам берет.

Башка тилдер сыяктуу, алгоритм тилинин да өзүнүн сөздүгү бар. Бул сөздүктүн негизин аткаруучунун командалар системасына кирип, бул же тигил алгоритмдин командаларын жазуу үчүн колдонулуучу сөздөр түзүшөт. Бул сөздүктүн жардамы менен **жөнөкөй** командаларды түзүүгө болот. Жөнөкөй командалар удаалаш аткарылат.

Жөнөкөй команда кыргыз тилинин толук же кыскартылган формадагы буйрук сүйлөмү катары берилет, керек болсо өз ичине формулаларды жана символдук белгилеништерди камтыйт. Мындан сырткары, маанилери жана колдонуштары биротоло бекиген сөздөрдүн кээ бири да алгоритм тилинде колдонулат. Бул сөздөр **кызматчы сөздөр** деп аталат. Алгоритмди жазууда алар өзгөчөлөнүп кыскартылып жазылат. Алгоритм тилинде жазылган алгоритм аталышка ээ болууга тийиш. Алгоритмдин аты анда чыгарылып жаткан маселелерге ылайыкталып коюлат. Алгоритмдин атын бөлүп көрсөтүү үчүн алардын астына **алг** (алгоритм) кызматчы сөзү жазылат.

Алгоритмдин атынан кийин анда колдонулуучу маалыматтардын тиби баяндалат жана алгоритмдеги аракеттер **башы** жана **аягы** кызматчы сөздөрүнүн ортосунда колдонулат. Жөнөкөй алгоритмде командалар удаа аткарылат. Эгер бир сапта бир нече команда жазылса, анда алар үтүрлүү чекит менен ажыратылып берилет.

Тартиби менен аткарылган алгоритмдин бир нече командаларынын удаалаштыгы **серия** деп аталат. Серия бир эле командадан да турушу мүмкүн. Алгоритм тилинде жазылган алгоритмдин жалпы түрү төмөнкүдөй болот: (Жопокой командалар).

алг алгоритмдин аты(маалыматтар тиби)
башы
алгоритмдин командалары(серия)

аягы

Мисалы: Y тин маанисин Y=(2x+3)(7x-5) формуласы менен эсептөө алгоритмин жазалы:

```
алг  у тин маанисин эсептөө(х,у - анык сан)
     арг х
     жый у
     башы
          х ти киргизүү
          у=(2*х+3)*(7*х-5)
          у ти чыгаруу
     аягы
```

СОСТАВДУУ КОМАНДАЛАР

Алгоритм тилинде эки составдуу команда: тармактуу жана кайталоо(цикл) командалары колдонулат. Бул командалардын жөнөкөй командадан айырмасы, алар ичине шартты камтып турат жана шартка ылайык, составдуу командага кирген бул же тигил команда аткарылат. Составдуу командаларды жазуу үчүн төмөнкү кызматчы сөздөр колдонулат: Эгер(эгерде), анда (анда), антп (антпесе), бүт(бүттү), азыр (азырынча), цб(циклдин башы), ца(циклдин аягы).

Тармактуу команда төмөнкү түрдө жазылат:

1. Толук формасы:

```
Эгер <шарт>
     анда <1-аракет>
     антп <2-аракет>
бүт
```

2. Толук эмес формасы:

```
Эгер <шарт>
     анда <аракет>
бүт
```

Кайталоо командасы төмөнкү түрдө жазылат:

```
Азыр шарт
     цб
     <аракет>
     ца
```

АЛГОРИТМДИН БЕРИЛИШ ЖОЛДОРУ (ЫКМАЛАРЫ)

Алгоритм, адамга түшүнүктүү болгон формаларда берилет. Эң кеңири тараган формаларына кыймыл аракет, сөз түрүндө, блок-схема, таблицалык, графикалык жана программа тилинде берилет.

Бул формаларда бир эле алгоритмди бири-бирин толуктап бирдиктүү колдонсо болот. Көпчүлүк учурда графикалык форма маселенин негизи катары колдонулат.

Мисалы: А жана В сандары берилген. Эгер А > В болсо, С = А – В ны эсепте, эгерде А <= В болсо, анда С = А + В ны эсепте. Бул мисалдын чыгарылышын алгоритмдин берилиш жолдору боюнча көрсөтөлү:

Сөз түрүндө берилиши – бул алгоритмди сөз жана сүйлөм менен сүрөттөйт. Бирок алгоритмдин эрежелери так сакталбайт. Алгоритм жазылышы өтө чоң формада болуп, түшүнүүгө кыйын болуп калат. Мисалы: Эгерде А В дан чоң болсо, анда А дан В ны кемит, жыйынтыгын С деп белгиле, эгерде А В дан кичине же ага барабар болсо, А менен В ны суммалап жыйынтыгын С деп алуу керек.

Сап түрүндө:

1) Саптар номерленет
2) Алгоритм саптардын өсүшү менен аткарылат.
3) Алгоритмде окуу, иштеп чыгаруу, логикалык байланыштар колдонулат.

Бул берилиште алгоритмдин так эместиги жоюлат, б.а алгоритм кадам менен аткарылат. Бирок кабыл алууга оор болот.

Мисалы:

1. Изделүүчү берилиштерди А жана Вны компьютердин эсине киргиз.
2. Эгерде А > В болсо, анада А – В ны кемитип эсепте.
3. Жыйынтыкты С га менчикте.
4. Эгерде А <= В болсо, анда А + В ны эсепте.
5. Жыйынтыкты С га менчикте.

Таблицалык түрү алгоритмдин таблицалык формада көрүнүшүн жана эсептоосу үчүн кызмат кылат. Изделүүчү жана чыгарылуучу белгилер графанын башына жазылат.

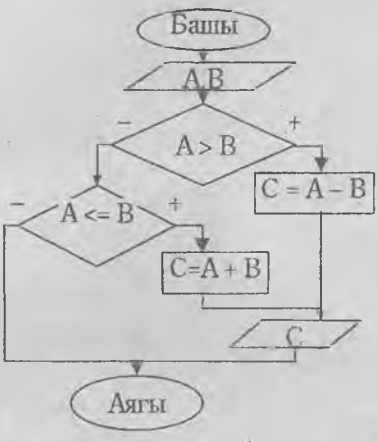
2-Мисал: С=К*Т формуласы аркылуу айлык акыны эсептөө талап кылынсын.

Фамилия	Иштеген күндүн саны, К	Тариф, Т	Айлык акы, С
Асанова	22	5,20	22*5,20
Эсеканова	14	3,80	14*3,80

Графикалык түрдө берилиши же блок-схема – бул алгоритмдин геометриялык фигуралардын жардамы менен көрсөтүлүшү. Мындай геометриялык фигура блок деп аталат. Схемада блок блокторду багытталган кесиндинин (стрелка) жардамы менен туташтырып турат. Мындай ар бир багытты тармак деп аташат. Алгоритм тилинде блок-

схема үчүн колдонулган геометриялык фигуралардын сүрөттөлүшү жана аткарган кызматы китептин "Тиркемелер" бөлүмүндө көрсөтүлгөн:

Блок-схемасы:



Алгоритм тилинде:

Алг < Мисал>
Башы
Кирг A, B
Эгер A>B **анда** C=A - B; 5 ке өт
Эгер A <= B **анда** C = A + B
5 **Чыг** C
Бүтүү
Бүтүү
Аягы

Алгоритмдин **программа түрүндө** берилиши:

Бейсик тилинде:

```
REM
INPUT A,B:IF A>B THEN C=A-B:GOTO 5
IF A<=B THEN C=A+B
5 PRINT C: END
```

Паскаль тилинде:

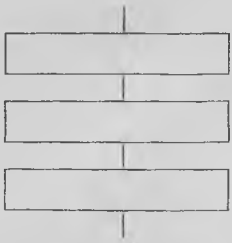
```
program esep;
var a,b,c:real;
begin read(a,b);
if a>b then c:=a-b else if a<=b then c:=a+b;
writeln('c=',c:8:3);end.
```

АЛГОРИТМДИ КЛАССИФИКАЦИЯЛОО
СЫЗЫКТУУ АЛГОРИТМ

Алгоритмдеги аткарылган операциялардын бири-бири менен байланышына жараша алгоритмди сызыктуу, тармактуу, циклдык деп бөлөбүз.

Жогоркудай үч типтүү алгоритмдин түзүлүштөрүнүн ичинен эң жөнөкөйү сызыктуу алгоритм болуп эсептелет. Эгерде алгоритм сызыктуу түзүлүшкө ээ болсо, анда ага кандай гана берилиш бербейли ага кирген кадамдар биринин артынан экинчиси удаалаш аткарылат. Бир сызыктуу алгоритм башка сызыктуу алгоритмден ага кирген блоктордун саны жана ага кирген блоктордун аткара турган кызматтары менен айырмаланышат:

Структурасы: Мисалы: Үй куруу алгоритми

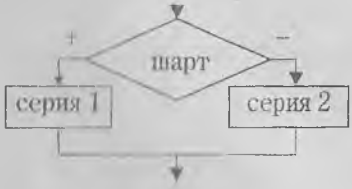


ТАРМАКТУУ АЛГОРИТМ

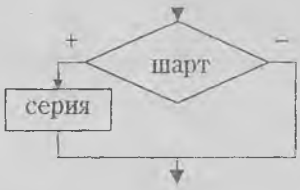
Алгоритмде бир же бир нече шарт орун алса **тармактуу алгоритм** деп аталат. Тармактуу алгоритм бир же бир нече логикалык шартты жана бир нече эсептелүүчү тармакты өз ичине камтыйт. Бир тармактын ичинде дагы эки жана андан көп тармак бар экендигин жолуктурабыз. Тармактуу алгоритмдин кайсы тармагы иштери же кайсы тармагы иштебеси койулган шартка жараша болот.

Мисалы, бир алгоритмге кирген инструкциялар жогортон төмөн карай иштеп келип эле кандайдыр бир кадамда шартка дуушар болсо, анда ал шарт эми кайсы топтогу кадамдар аткарыларын аныктайт. Алгоритм өзүнүн татаал же жөнөкөй экендигине карабастан, аның структурасына көз карандысыз дайыма бир эле "аягы" болот. Ар бир тармакты эсептеп отуруп, биз бир эле "аягы" деген блокко чыгабыз. Берилиш структуралары төмөнкүдөй(блок-схемада):

а) толук формасы:



б) толук эмес формасы

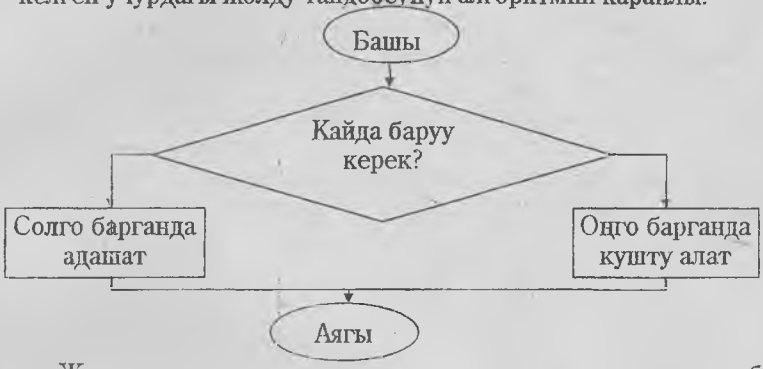


1-Мисал.

Алг Маршрут
Башы **Эгер** автобус келсе **анда** автобуска түшүү
антп жөө басуу
бүттү
Аягы

Бул алгоритмде эгер автобус болсо, анда автобуска түшөбүз, эгер автобус келбей калса, жөө басууга туура келет.

2-Мисал. “Алтын Куш” жомогундагы баланын кара таштын жанына келген учурдагы жолду тандоосунун алгоритмин карайлы:



Жалпысынан караганда тармактын саны экиден көп болушу мүмкүн.

3-Мисал

Алг Галлактика

Башы

эгер спутниги болсо **анда** **эгер** **Күндөн** алыс болсо

анда Ал планета Уран

антп Ал планета Жер

антп **эгер** **Күнгө** жакын болсо

анда Ал планета Меркурий

антп Ал планета Венера

бүттү

бүттү

бүттү

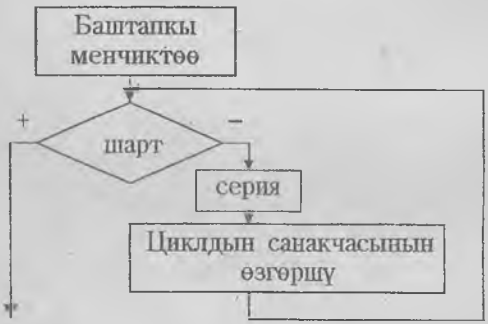
Аягы

Бул алгоритмде бир нече шартка жараша планеталардын түрлөрү аныкталат.

ЦИКЛДЫК АЛГОРИТМ

Цикл – бул бир нече жолу кайталануучу алгоритмдин бөлүгү.
Циклдын параметри – циклга ар бир киргенде жаңы мааниге ээ болгон өзгөрмө.

Эгерде алгоритмге кирген бир же бир нече удаалаш амалдар бир же бир нече жолу аткарылса анда мындай алгоритмдер **циклдык алгоритм** деп аталат. Циклдин кайталануучу бөлүгү **циклдин телосу** деп аталат. Блок-схемада көрсөтүлгөн циклдык алгоритм төмөндөгү-дөй структурад а берилет:



Төмөнкү мисалдарды карайлы.

1-Мисал. 1 ден 10 го чейинки натуралдык сандарды квадратка көтөрүү алгоритми.

```
алг сандарды квадратка көтөрүү
    башы
        цб   x=1 ден 10 ке чейин
             x=x*x
             чыг x
        ца
        бит
    аягы
```

Мында өзгөрмө x тин алгачкы мааниси 1, акыркы мааниси 10 жана ал улам 1 ге өсүп отурат. Цикл 10 жолу аткарылат, б.а. x тин ар бир маанисинде ошол сандын квадраты эсептелинип чыгарылат. Мисалы: эгер x=5 болсо, анда 5 тин квадраты 25, ж.б.

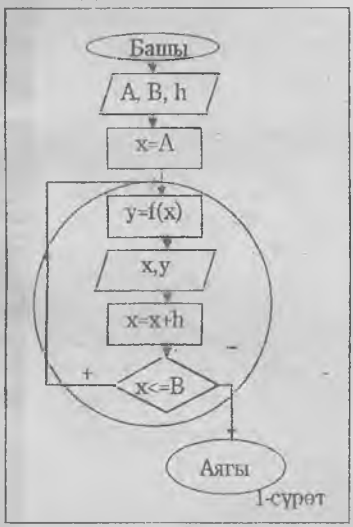
2-Мисал

```
алг тарбиялоо
  башы
  азырынча тартип бузуучу мойнуна алганча
     цб  тартип маселеси боюнча
         ангемелешүүнү жүргүзүү
```

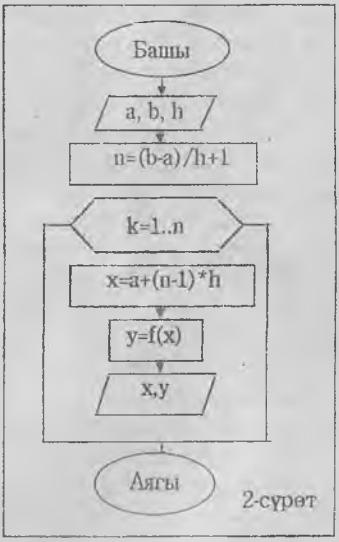
ца
бүтүү
аягы

Бул алгоритмдин жыйынтыгы тартип бузуучу мойнуна алып, түшүнгөнгө чейин аңгемелешүү улантыла берет. Төмөнкүдөй формулировкаланган бир өзгөрмөлүү функцияны табуляциялоонун маселеси да циклдык алгоритмдин мисалы боло алат.

А дан В га чейинки h туракту кадамы менен өзгөртүлгөн y=f(x) функциясынын маанисин эсептөөлөр төмөнкүдөй жүргүзүлөт. Баштапкы x =a үчүн y тин мааниси чыгарылат. x тин мааниси h кадамына чоңойот жана ал үчүн y тин жаңы мааниси эсептелет. Бул этап x(b) нын маанисинен ашканча кайталана берет. Алгоритмдин схемасы 1-сүрөттөгүдөй болот.



1-сүрөт



2-сүрөт

Бул убакта циклдин телосу n=(b-a)/h+1 жолу кайталанары шексиз. Муну эске алуу менен маселенин кайталоолорунун саны белгилүү болгон цикл түрүндө берсе болот (2-сүрөт).

Каалагандай цикл өзүнүн ичине бир же бир нече циклдерди камтышы мүмкүн. Алгоритмдердин бардыгы сызыктуу, тармактуу жана циклдик структуралардын ар кандай комбинацияларына келтирилет.

МАССИВДИ ИШТЕТҮҮДӨ КОЛДОНУЛУУЧУ АЛГОРИТМДЕР

Жогорудагы алгоритмдин түрлөрүн карап чыкканыбыздан кийин массив түшүнүгүн жана массивдерди иштетүүдө алгоритмди кандай пайдалануу керек экендигин карап кетели.

Массив – белгилүү ысымга ээ, индекстери боюнча иреттелген бир тектеги элементтердин көптүгү.

Мисалы: Бактын бир бутагында тизиле конуп отурган чабалекейлердин көптүгү, шаардын калкынын саны, класстагы окуучулардын саны, класстагы парталардын саны. Эгерде биз мисалга класстагы окуучулардын фамилияларынан түзүлгөн массивди карай турган болсок, ал төмөндөгүдөй болот. Массивдин ысмы –"Окуучулар". Класстык журналдагы тизме боюнча 1-турган окуучунун фамилиясы массивдин 1-элементи, 2-турган окуучунун фамилиясы массивдин 2-элементи, 3-турган окуучунун фамилиясы массивдин 3-элементи ж.б. акыркысы массивдин акыркы элементи болуп саналат.

Ошентип, массив тизмектелген элементтердин көптүгү болуп эсептелет, ар бир элементи бири-биринен катар номери жана ар бир элементке тиешелүү мааниси менен айрымаланышат. Массивди жазууда массивдин ысмы, андан кийин анын тиешелүү элементине таандык катар номери(индекси) тегерек кашаага алынып жазылат.

Мисалы, Окуучулар(1) деген жазуу "Окуучулар" деген массивдин 1-элементи, Окуучулар(5) деген жазуу массивдин 5-элементи. Санак болсо 1 деген сандан баштап саналгандыктан, класста бардыгы 30 окуучу окуган болсо, анда массивдин биринчи элементинин мааниси ошол окуучунун фамилиясына, ал эми акыркы элементинин мааниси 30-окуучунун фамилиясына туура келет. Жазуу төмөндөгүдөй болот.

Окуучулар(1)="Асанов"
Окуучулар(2)="Бекенов"
...
Окуучулар(30)="Эшимов"

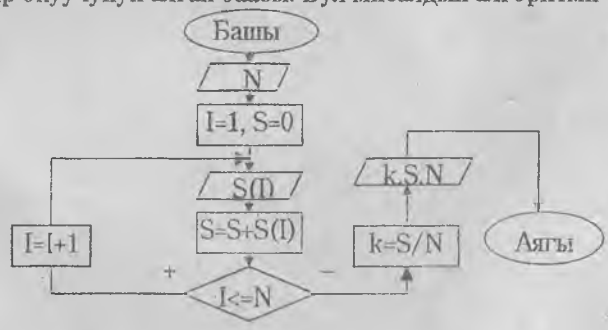
Эгерде массив бир эле индекс менен аныкталса, анда аны бир өлчөмдүү массив; массив эки индекстен турса б.а. узуну туурасы менен мүнөздөлсө анда аны эки өлчөмдүү массив; ал эми массив узуну, туурасы жана бийик-тиги менен аныкталса, анда аны үч өлчөмдүү деп атайбыз ж.б. Бир өлчөмдүү массивке мисал: катарга тизилип турган спортсмендер, журналдагы окуучулардын фамилияларынын тизмеси, бир жылдагы күндөрдүн санын мисалга алсак болот. Эки өлчөмдүү массивке мисал катары: класстагы парталардын жайгашышы, театрдагы отургучтардын жайгашышы, журналдагы баалардын коюлушу ж.б. көптөгөн мисалдарды алсак болот. үч өлчөмдүү массивке, мисалы, Рубиктин кубигин алсак болот.

Эми жогорудагы "Окуучулар" деген массивдин алгоритмин түзөлү: Мында N сандагы окуучулар жана N = 30 болсо, анда окуучулардын тизмесин чыгаруу алгоритмин блок-схема түрүндө берсек, он жакта көрсөтүлгөндөй болот. Мында N=30 болсо, анда түзүлгөн алгоритм төмөнкүдөй иштейт. I = 1(катар номер) болгондо Z(1) = "Асанов" деген маанини алат, андан кийин шарт текшерилет жана ал канааттандырылгандыктан I = I + 1; I = 1 + 1; I = 2 маанисин алат. Ошентип цикл I = 30 болгончо аткарыла берет. Мында I санакчынын да, катар номердин да маанисин туюндурат. Жыйынтыгында окуучулардын тизмеси чыгарылат.

2-мисал: Класста 15 окуучу болсун дейли. Алардын информатика сабагынан чейрек боюнча алган бааларынын орточо маанисин аныктай турган алгоритмди түзөлү. Бул алгоритмдин максатына жетүү үчүн биз окуучулардын алган бааларын кошуп, келип чыккан жыйынтыкты окуучулардын санына бөлөбүз. Ал төмөнкү формула менен аныкталат.

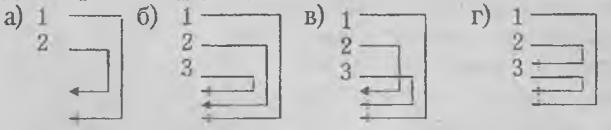
$$k = \frac{\sum_{i=1}^{n} S_i}{n}$$

мында n = 15; S_i – массивдин элементтерин түзөт (i =1,2, ..., 15), б.а. ар бир окуучунун алган баасы. Бул мисалдын алгоритми төмөнкүдөй:



КИЙИШТИРИЛГЕН ЦИКЛДАР

Жогоруда цикл түзүүчү бир нече мисалдарды карадык. Айрым учурда цикл жасоочу алгоритмдин ичинде да цикл жасоочу алгоритм болуп калышы мүмкүн. Мындай учурда бири-бирине кийгизилген циклдерге ээ болобуз. Төмөнкү схемада циклдердин бир нече түрү көрсөтүлөт:

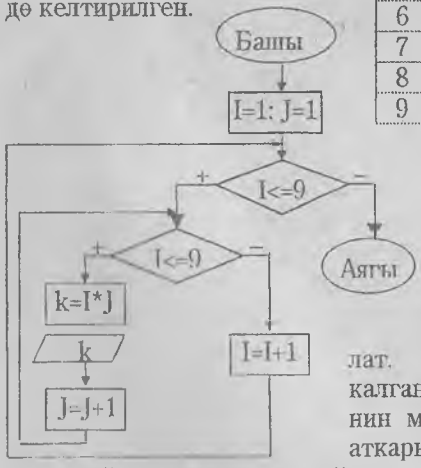


а) пунктунда 1-цикл сырткы, 2-цикл ички болот. б) пунктунда бири-бирине кийиштирилген 3 цикл бар. в) пунктунда ката түзүлгөн цикл. г) пунктунда 1-цикл сырткы анын ичиндеги 2 цикл бири-биринен көз карандысыз түзүлгөн ички циклдер.

Мисалы: Пифагордун таблицасын эсептөөнүн алгоритмин түзөлү.

1	2	3	4	5	6	7	8	9
2	4	6	8	10	12	14	16	18
3	6	9	12	15	18	21	24	27
4	8	12	16	20	24	28	32	36
5	10	15	20	25	30	35	40	45
6	12	18	24	30	36	42	48	54
7	14	21	28	35	42	49	56	63
8	16	24	32	40	48	56	64	72
9	18	27	36	45	54	63	72	81

I –мамыча боюнча индекстин өзгөрүүсү J – сап боюнча индекстин өзгөрүүсү. Бул мисалга түзүлгөн алгоритм төмөндө келтирилген.



Мында I=1 жана J=1 болгондо экөөнүн көбөйтүндүсү 1 деген сан болору көрүнүп турат. Андан кийин шарт текшерилет жана ал шартка жооп бергендиктен J нын мааниси бирге чоноёт да көбөйтүү аткарылат. Бул процесс шарт аткарылбай калганга чейин аткарылып, андан кийин I нин маанисин бирге чонойтуп көбөйтүү аткарылат жана I нин мааниси 9 -деген санга чейин аткарылып, жыйынтыгында таблицадагы сандарды алабыз, башкача айтканда көбөйтүүнүн таблицасы чыгат. Мында I боюнча өзгөртүү–сырткы цикл, ал эми J боюнча өзгөртүү–ички цикл деп аталат.

ЖАРДАМЧЫ АЛГОРИТМ

Бир эсепке ылайык алгоритм түзгөн учурда анын айрым бир жерлеринде бир түрдүү кадамдарды улам кайталоого туура келип калат. Ошондуктан, мындай абалды жеңилдетиш үчүн кайталапуучу кадамдарды жакшылап карап чыгып, ага өзүнчө алгоритм түзөбүз.

Мисалы: Ашканадагы окуучунун аткарган милдети

Жатаканада жаткан окуучу күнүгө ашканадан үч маал тамак ичет. Ар бир күнү тамак ичилип бүткөндөн кийин окуучу төмөнкү милдетти кайталашы зарыл: стол үстүндөгү идиштерди жыйнайт, столду сүртөт, стулдарды ордуна жылдырат. Эми биз бул берилгендер боюнча алгоритмин түзгөнгө аракет кылып көрөлү:

1) Эртең мененки тамактануу 2) Ашканага келүү
3) Даяр тамакты ичүү
4) стол үстүндөгү идиштерди жыйноо, столду сүртүү, стулдарды ордуна жылдыруу
5) Түшкү тамактануу 6) Ашканага келүү 7) Даяр тамакты ичүү
8) стол үстүндөгү идиштерди жыйноо, столду сүртүү, стулдарды ордуна жылдыруу
9) Кечки тамак 10) Ашканага келүү 11) Даяр тамакты ичүү
12) стол үстүндөгү идиштерди жыйноо, столду сүртүү, стулдарды ордуна жылдыруу

Жогоруда көрсөтүлгөн алгоритмде 4,8,12-саптагы командалар окшош жана алгоритмдин аткарылышында буларга кайрылууну биз бир нече жолу аткарып жатабыз. Алгоритмде бул командаларга уламдан улам кайрылып отуруу алгоритмди татаалдаштырат. Мындай ыңгайсыздыкты жоюу үчүн биз жардамчы алгоритмдер деп аталга түшүнүк киргизебиз.

Башка алгоритмдин составында толугу менен пайдалануучу алгоритм **жардамчы алгоритм** деп аталат. Жардамчы алгоритм алгоритмдердин структурасын жөнөкөй жана түшүнүктүү кылат жана башка колдонуучулардын идеяларын алгоритмге кошуп иштетет. Эми жогоруда берилген "Ашканада окуу чунун аткарган милдети" алгоритмин жардамчы алгоритмди колдонуп карап көрөлү:

1) Эртең мененки тамактануу
2) Ашканага келүү
3) Даяр тамакты ичүү
4) Окуучунун милдети
5) Түшкү тамактануу
6) Ашканага келүү
7) Даяр тамакты ичүү
8) Окуучунун милдети
9) Кечки тамак
10) Ашканага келүү
11) Даяр тамакты ичүү
12) Окуучунун милдети

Окуучунун милдети: стол үстүндөгү идиштерди жыйноо, столду сүртүү, стулдарды ордуна жылдыруу

Мисалы:

Алг Үч сандын чоңун табуу (анык a,b,c,y)

```
Арг a,b,c
Жый y
    Башы
    аныk z
        Эсч(a,b,z); Эсч(z,c,y)
    Аягы
```

Демек, алгоритмдин кайталануучу бөлүгүн бир жолу эле жазып, керектүү жерде ошого кайрылып турууга боло тургандыгы жогорудагы мисалдан көрүнүп турат. Ошондой эле жардамчы алгоритмдин жардамы менен татаал алгоритмди жөнөкөйлөтүүгө болот.

2-мисал. Киргизилген а нын маанисин төмөнкү туюнтмаларды аткаруу үчүн колдонууну жардамчы алгоритм менен иштетүүнүн алгоритмин карайлы.

Бул фрагментте а санынын киргизилген маанисинде X=2+a/5 туюнтмасынын маанисин эсептөө жана а нын 2.5 тен кичине маанисин текшерип бул шарт аткарылса $X=a^2$ туюнтмасы аткарыла тургандыгынын алгоритми жардамчы алгоритмде эсептелди жана каалагандай а нын маанилеринде, каалагандай туюнтмаларды эсептөөдө бул алгоритмди колдонууга болот.

Жогоруда берилген алгоритм жөнүндө түшүнүк компьютерде программалоону үйрөнүүдө алгачкы кадам болуп эсептелет.

II БӨЛҮМ

TURBO PASCAL ПРОГРАММАЛОО ТИЛИ

1. ПАСКАЛДЫН ПРОГРАММАЛЫК ЧӨЙРӨСҮ

АЛГАЧКЫ КАДАМ

Бул бөлүм азыркы учурда көпчүлүккө таанымал болгон Turbo Pascal (мындан ары жөн эле Паскаль деп атайбыз) программалоо системасына арналат. Эгер сиз буга чейин бул система жөнүндө кабардар болбосоңуз, анда аталышы сөзсүз кызыктырат. Turbo – Borland International Inc(США) фирмасынын соодалык маркасы, ал эми Паскаль кеңири тараган программалоо тили. Демек Turbo Pascal – Borland фирмасы тарабынан түзүлгөн программалоо тили.

Паскаль программалоо тили негизинен IBM (International Business Machines corporation) фирмасынын IBM PC(Personal Computer – жеке компьютер), IBM PC/XT (Extention Technology – технологиянын кеңейтилиши) IBM PC/AT (Advanced Technology - технологиянын келечектүүлүгү) жана IBM PS/2 (Personal System – жекече система) компьютерлеринде кеңири колдонулат.

Бул компьютерлер 80-жылдардан баштап пайда болуп, кыска мөөнөттүн ичинде таанымалдуулукка кандай жетсе, Паскаль тили дагы ошондой таанымалдуулукка жетти десек жаңылбайбыз. Паскаль программалоо тилинин тарыхынан бир тутум кабар ала турган болсок, 1968-1971-жылдары Швейцариянын Цюрих информатика институтунда Никлаус Вирт тарабынан иштелип чыккан. Алгачкы максаты, программалоону окутуучу программалоо тили болгон. Азыркы учурда болсо эң кеңири тараган обьективдүү программалоо тилдеринин катарына киргени сизге маалым.

Биз Паскаль (Франциянын улуу математиги жана философу Блез Паскалдын(1623-1662) наамына берилген) тилин баштоодо алгач ирет эске жүктөө милдети турганын эстен чыгарбоо керекпиз б.а. биздин

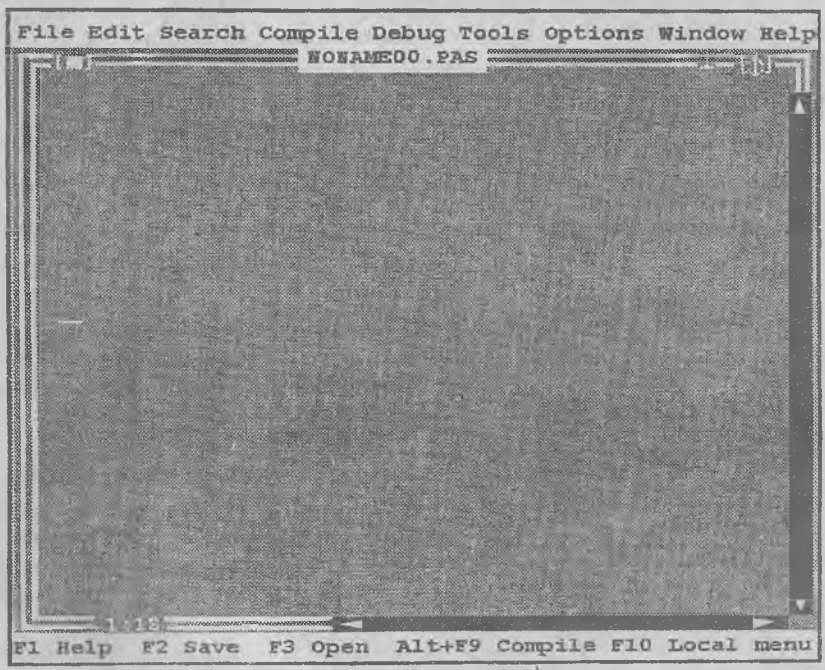
учурдагы дискте жайгашкан Паскалдын файлдарын камтыган каталогду активдештирүү менен `turbo.exe` файлын жүктөө жетиштүү.

Мисалы: Биздин каталог TP болсо

```
C:\cd TP
C:\TP\turbo.exe
```

деп буйрук сапчасына киргизебиз.

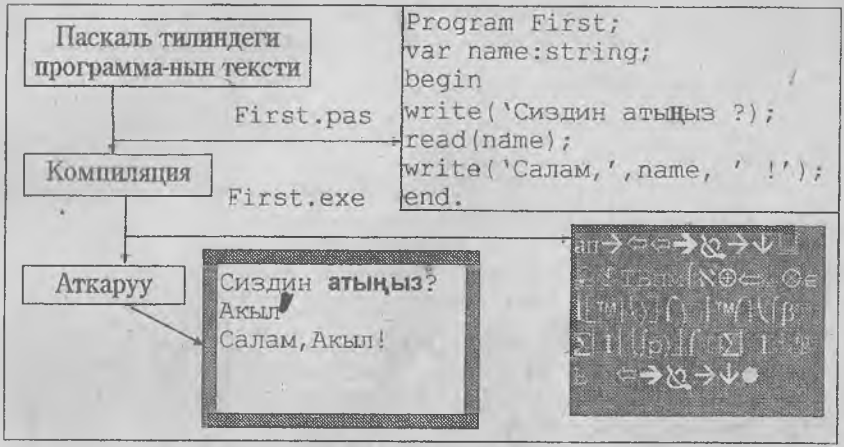
Алгачкы кадамдын натыйжасы ийгиликтүү болсо, анда биздин экранга 3-сүрөттөгүдөй көрүнүштү камтыган Паскаль системасынын чөйрөсүн байкайбыз.



3-сүрөт

Ал эми бул жерден баса белгилөөчү кеп кабыл алынган системадан да чыгууну унутпоо керек. Сиз иш максатыңызга жеткен соң бул системадан чыгууну аткаруу үчүн `Alt` баскычын басып турып кое бербестен `X` латын тамгасын камтыган баскычты, б.а. `Alt+X` баскычтарынын комбинациясынын жардамы менен басабыз. Система жүктөлгөн соң, биз бир нече структураны камтыган чөйрөгө туш болобуз. Алар буйрук тандоо сапчасы, программанын текстин камтыган редактор жана маалымдоо сапчасынан турат.

Бул системалык чөйрөдө иштөө бир нече этаптарды өз ичине камтыйт. Алар томонку схемада келтирилген:



Программа чөйрөсүндөгү аткарылган иштин жыйынтыгы болуп программанын тексти(кеңейтилиши pas болгон файлдар) жана бул чөйрөгө көз карандысыз аткарылган файлдар (кеңейтилиши exe) эсептелинишет.

Ошондой эле маалымдоо сапчасында көрсөтүлгөн функционалдык баскычтар менен иштөө ишти тездетет. Функционалдык баскычтар Паскаль системасынын башкаруу кызматын аткарат жана алардын белгилениши F1 ... F10. Баскычтар клавиатуранын жогорку бөлүгүндө жайгашкан. Мындан сырткары Alt(Alternati-ve – кошумча), Ctrl (ConTroL – башкаруу) жана Shift(shift - жылышуу) баскычтарын колдонуу менен башкаруу буйруктарынын санын көбөйтөбүз. Демек биз жогорку үч баскычтын бирөөнү басып туруп коё бербестен функционалдык баскычтарды же жөн баскычтарды бассак, тиешелүү болгон буйруктар аткарылат. Алар төмөндө келтирилген(ошондой эле китептин "Тиркемелер" бөлүгүндө толугураак маалымат берилген):

F1 – жардам
F2 – оңдолуучу текстти дисктеги файлга жазуу
F3 – дискте жайгашкан текстти окуу
F4 – жөндөө режиминде колдонулат
F5 – жөндөө терезесин толук экранга чоңойтот; кайра басууда алгачкы учуруна алып келет
F6 – оңдоо терезесин жөндөөчү терезеге которот
F7 – жөндөө режиминде колдонулат
F10 – башкы менюга чыгуу
Alt+F9 – программаны компиляциялоо
Ctrl+F9 – программаны компиляциялоо жана аткаруу
Alt+F5 – редактордун терезесин жыйынтыктар чыгуучу экранга которот

Alt+X – Паскаль чөйрөсүнөн чыгуу

3-сүрөттө көрүнүп тургандай башкы меню 10 негизги пункттан турат:

File – файлдар менен иштөө

Edit – текстти редактирлөө, анын бөлүктөрүн көчүрүү жана жылдыруу

Search – керектүү текстти издөө, сөздөрдү алмаштыруу, издөөнүн башка инструменттери

Run – программаны аткаруу, кадам боюнча жөндөө

Compile – программаны компиляциялоо

Debug – жөндөө программасынын инструменттери

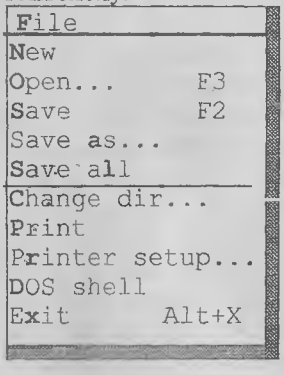
Tools – чөйрөнүн инструменттери

Options – программа чөйрөсүн жөнгө салуу

Window – жөндөө программасынын терезелерин башкаруу

Help – жардам

Башкы менюга кайрылуу F10 баскычынын жардамы менен жүргүзүлөт. Эми бул менюлардын кээ биринин аткарган милдеттерине токтололу.



File менюсунда сиздерге тааныш болгон командалар жайгашкан:

New – жаңы файл түзүү

Open – мурда түзүлгөн файлды ачуу

Save – дисктеги активдүү терезеден активдүү каталогго сактоо

Save as – файлдын атын жана багытын көрсөтүп дискке сактоо

Save all – бардык файлдарды сактоо

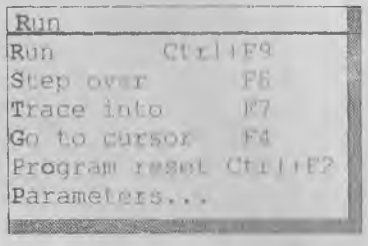
Change dir ... – жумушчу каталогду өзгөртүү

Print – файлды кагазга чыгаруу

Print setup ... – принтерди жөнгө салуу

DOS shell – DOS терезеси

Exit – Паскаль чойрөсүнөн чыгуу



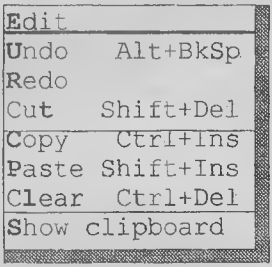
Run менюсу төмөндөгү командаларды өз ичине алат:

Run – программаны аткаруу

Step over – программаны процедурага кирбестен саптар боюнча аткаруу

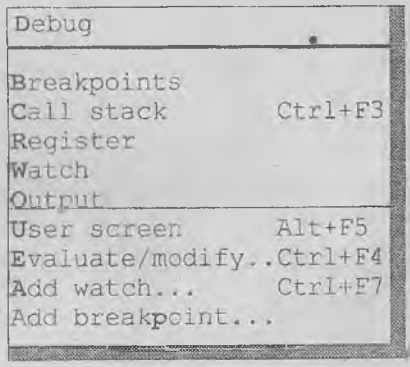
Trace into – программаны саптар боюнча аткаруу

Go to cursor – курсор турган сапка чейин программаны аткаруу
Program reset – программаны жөндөөну токтотуу .
Parameters... – программалоо чөйрөсүндөгү программанын параметрлерин берүү



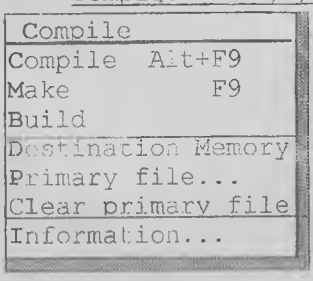
Edit менюсунда төмөнкүдөй командалар жайгашкан:
Undo – акыркы аракетти кайтаруу
Redo – Undo командасы аткарган аракетке каршы аракет
Cut – буферге жайгаштыруу
Copy – буферге көчүрүү
Paste – буфердеги текстти жайгаштыруу
Clear – өчүрүү
Show clipboard – тексттин бөлүктөрүн көчүрүүдө жана жайгаштырууда колдонулуучу атайын буфер

Debug менюсунун командаларынын аткарган кызматы:



Breakpoints – контролдук чекиттер
Call stack – стек-терезеси
Register – регистр-терезеси
Watch – жөндөө терезеси
Output – программанын терезеси
User screen – колдонуучунун экраны
Evaluate/modify... – өзгөрмөнү көрүү жана өзгөртүү
Add watch... – жөндөө терезесине кошуу
Add breakpoint... – контролдук чекиттерди кошуу

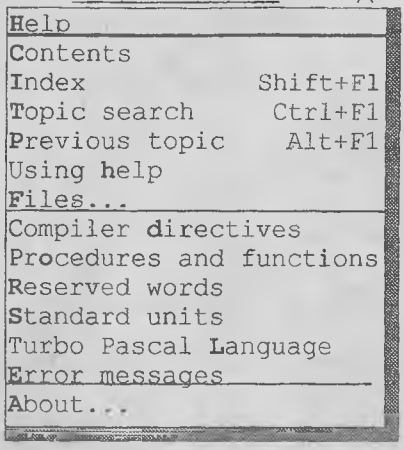
Compile менюсунун командаларынын аткарган кызматы:



Compile – программаны компиляциялоо
Make – Программаны иштетпей текшерүү
Build – бардык файлдарды толук компиляциялоо
Destination Memory – корголгон режим үчүн компиляциялоо
Primary file... – негизги файлды компиляциялоо
Clear primary file – негизги файлдын жазылышын тазалоо

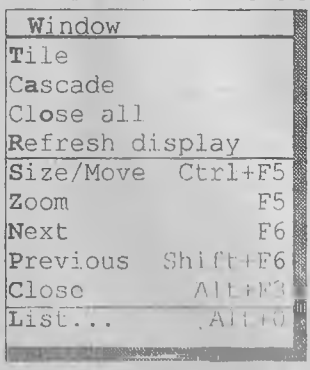
Information... – информация

Help менюсунун командаларынын аткарган кызматы:



Contents – Паскалдын негизги бөлүктөрү боюнча жардам
Index – программалоодо колдонулган бардык командалар боюнча алфавиттик тартипте берилүүчү жардам
Topic search – бул деле Index сыяктуу
Previous topic – жардам
Using help – жардам
Files...– жардамчы файлдарды инициализациялоо
Compiler directives – жардам каталогу
Procedures and functions– процедуралар жана функциялар боюнча жардам
Reserved words – кызматчы сөздөр боюнча жардам
Standard units–стандарттуу модулдардын иштеши боюнча жардам
Turbo Pascal Language – Турбо-Паскаль программалоо тили боюнча жардам
Error messages – программанын иштешине системалык жардам
About... – Турбо-Паскаль жөнүндө

Window менюсунун командалары төмөнкү кызматтарды аткарат:



Tile – экранды тик бурчтуу терезелерге бөлүү
Cascade – терезелердин бири-биринин үстүнө жайгашышы
Close all – баарын жабуу
Refresh display – экранды жаңыртуу
Size/Move – өлчөм/которуу
Zoom – терезени максималдаштыруу
Next – кийинки терезе
Previous – мурунку терезе
Close – терезени жабуу
List... – терезелердин тизмеси

АЛГАЧКЫ ПРОГРАММА

Келиңиз анда биздин экранга кандайдыр бир кабар чыгара турган эң жөнөкөй программа түзөлү. Бул программа сиздин Паскаль чөйрөсүндөгү тушоо кесүү программаңыз болсун.

```
Program My_program;
Begin
    Write('Тушоо кесүү');
End.
```

Мында `Program` кызматчы сөзү программанын башталышын түшүндүрөт, андан соң программанын ысмы сөзсүз түрдө берилиши керек. `Begin ... end.` **программалык кашаа** деп аталат.

ТИЛДИН АЛФАВИТИ ЖАНА СТРУКТУРАСЫ

Тилдин баштапкы алфавит, алфавиттен сөз, сөздөрдөн кандайдыр бир түшүнүк бере турган сүйлөм жана сүйлөмдөрдүн жыйындысы болот эмеспи. Ал эми программалоо тили дагы ушул сыяктуу алфавиттен (символдордон) туруп алардын негизинде берилген атайын буйруктар, кызматчы сөздөр, операторлор жана функциялар куралат. Паскаль программалоо тилинин алфавитин негизинен үч бөлүккө бөлүп караса болот: тамгалар, сандар жана атайын символдор:

- Тамгаларга – латын тамгалары A – Z, a – z жана орус тамгалары А – Я, а – я. Орус тамгалары комментарийлер жана ар кандай түшүндүрмөлөргө гана колдонулат;
- Сандарга – араб цифралары: 0,1,2,3,4,5,6,7,8,9;
- Атайын символдор:

Атайын символдор да үч бөлүккө оз ара бөлүнүп кетет:

- Арифметикалык операциялар үчүн (+, – , * , /);
- Салыштыруу операциялары үчүн (<, >, =, >=, <=);
- Ажыраткыч жана кошумча (. , : ; ! # $ ж.б.);

Программа – компьютер үчүн берилген маселенин алгоритмин иштетүүнү камсыз кылган аракеттердин иреттүү удаалаштыгы. Паскаль тилинде программанын жазылышы `Program` кызматчы сөзү менен башталып, `end` кызматчы сөзү жана чекит менен аяктайт. Бул эки кызматчы сөз камтыган эки бөлүк бар: маалыматтарды сактоочу чоңдуктарды баяндоо жана аракеттердин тобу(операторлор):

```
Program <ысмы>;   {Программанын аталыш_аймагы}
Баяндоо бөлүгү ;
Begin {Аракеттер тобунун башталышы.
       Программалык кашаны ачуу}
```

```
Аракеттердин тобу;
End.{Программанын аягы. Программалык кашанын жабылышы}
```

Программанын ысмынын берилиши зарыл жана чоңдуктарды баяндоо бөлүгү төмөнкү кызматчы сөздөрдөн турат: `Uses,label, const,type,var,Procedure` жана `Function`. Булардын ар бири не кайрылуунун кереги жок, себеби убагы менен ар бирин карап чыгабыз. Аракеттер тобу `begin` кызматчы сөзү менен башталат жана ар бир көрсөтмө буйрук аяктаганда үтүрлүү чекит менен бекемделет. Программанын аяктаганда `end` кызматчы сөзү жана чекит менен аяктайт.

МААЛЫМАТТАРДЫН ТИБИ

Константа – программанын иштетүүдө өз маанисин өзгөртпөгөн туракту чоңдук жана ал `Const` деп белгиленет.

Өзгөрмө – программанын иштетүүдө кайрылуу жасоо шарттарына жараша өз маанисин өзгөртүүчү чоңдук жана ал `VARIABLE(var)` деп белгиленет. Берилиштер – формалдуу түрдө берилген информация, ал информацияны сактоо, иштетүү жана берүү мүмкүнчүлүгүн камсыз кылат жана `data, information` деп белгиленет.

Паскалда берилген туракту уубу, өзгөрмөбү, функция же туюнтманын маанилериби өз типтерине жараша объект деп берилип, аткаруу жана кабыл алуу маалыматтарына жараша объектинин тибин берет. Объектинин тиби ага колдонууга боло турган операциялардын тобу, ошондой эле бул операциялардын аткарылышынын жыйынтыгынын тибин камтыган көптөгөн маалыматтарды аныктайт. Тиби төмөнкүдөй таблица менен берилет:

Жөнөкөй		Татаал	
Бүтүн	`Integer, byte, word`	Жазылыштар	`Record`
Логикалык	`Boolean`	Көптүк	`Set`
Символдук	`Char`	Файл	`File`
Анык	`Real,Single,Double, Extended`	Сантык	`String`
Массивдер	`Array`	Көрсөткүчтөр	`Pointer`

ЫСЫМДАР(АТАЛЫШТАР)

Программалоодо кеңири колдонуучу сөз "ысым" "идентификатор" деген сөз менен алмашылып айтылат. Бул сөз латын сөзүнөн `identifico` – теңдештирүү деген түшүнүктү берет.

Программа түзүүдө берилген чоңдуктардын ысымын аныктоо, б.а. ысым ыйгарууга туура келет. Ал эми ысым (идентификатор) берүүнүн зарыл шарттары болуп төмөнкүлөр эсептелинет:

- Идентификатор дайыма тамга менен башталат, андан соң сандар же белгилөө сызыкчасы менен берилет;
- Бошток жана атайын символдор камтылбайт;
- Идентификатордун узундугу 63 символго чейинки маанини алат.

Мисалы: Туура берилиштери: `n135,a,data,mas,n_1,al_001`, ж.б.у.с.;

Туура эмес берилиштери: `325ak`, (сан менен башталган), `bb#20` (атайын символдор камтылган), `a 121` (бошток камтылган), `begin` (кызматчы сөз камтылган).

Экинчи программа. Биз Паскалда программалоону жакшы өздөштүрбөсөк да бир аз жеңилдетилген программаларды түзүүгө жетиштик. Демек биз программалоодо берилүүчү комментарий түшүнүгүн алабыз. Комментарий адам үчүн түшүнүк берүүчү объект, бирок программанын иштешине жолтоо болбойт.

Мисалы: Эки сандын суммасын табуу маселесин карасак.

```
Program Second; {Программанын аталыш аймагы,
                 Second - программанын ысмы}
Var a,b:integer;{кошуучу жана кошуулуучу сандар,
                 тиби - бүтүн}
c:integer; {натыйжасы}
Begin {программанын башы}
Read(a,b); {a,b сандарынын маанисин киргизүү}
c:=a+b; {Ыйгаруу}
Write(c); {натыйжаны экранга чыгаруу}
End. { Аягы}
```

Комментарийлерди "{", "}" кашаалардын ичине жазып, программанын каалаган жерине жайгаштырсак болот. Программанын негизги максаты жогоруда берилген. Мында "a" жана "b" идентификаторлору баштапкы маани катары сиз тараптан киргизилет жана булардын суммасын эсептеп "c" идентификаторуна ыйгаруу ишин аткаруу менен программанын аягына чыгат.

Ошондой эле бул жерден сиздер `read` жана `write` операторлорун байкаган чыгарсыздар. `Read` оператору – маалыматты компьютердин эсине киргизүү, ал эми `write` оператору – маалыматты дисплейдин экранына чыгаруу ишин аткарат.

АРИФМЕТИКАЛЫК АМАЛДАР ЖАНА ТУЮНТМА

Паскалда программа түзүп жатып, типтери `real` жана `integer` болгон өзгөрмөлөрдүн үстүнөн бардык төрт арифметикалык амалды аткара алабыз.

+	кошуу	/	бөлүү
-	кемитүү	div	бүтүн бөлүгүн алуу
*	көбөйтүү	mod	калдыгын алуу

Көңүл бургула! Паскалда жогорку тартиптеги даражага көтөрүү мүмкүнчүлүгү каралбайт.

Арифметикалык амалдардын аткарылуу мисалдарын карап көрөлү: **Туура:**

```
Var a,b:integer;
    r:integer;
    s:integer;.....;
    r:=a div b; {a=7, b=2 болгондо r=3}
    r:=a mod b; {a=7, b=2 болгондо r=1}
    s:=a*b;
    s:=a*b-a div b;
```

Туура эмес:

```
Var a,b:real;
    r:integer;.....;
    r:=a div b; {аныk сандар үчүн div операциясын
                 иштетүүгө болбойт}
    r:=a mod b; {mod операциясы бүтүн сандарга
                 карата колдонулат}
```

Көңүл бургула! Өзгөрмөлөрдүн тибин баяндоодо абдан так болуш керек. Өзгөрмөлөрдүн тиби төмөнкүлөрдү аткарууга жол берет:

1) өзгөрмөнүн ички көрсөтүлүшүнүн узундугун берүүгө;
2) программадагы бул өзгөрмөнүн үстүнөн болгон аракеттерди контролдоого. Өзгөрмөлөрдүн типтерин алгачкы контролдоо компиляциялоо этабында жүргүзүлөт. Математикалык туюнтманын жазылышы программалык текстте, б.а. көбөйтүү жана бөлүү амалдарында бир аз өзгөртүүгө учурайт.

Математикалык жазылышы	Паскаль тилинде жазылышы
a b	a/b(a,b - аныk) a div b(a, b - бүтүн, жоон бүтүн)
$\frac{(a+b)\cdot c}{2d}$	(жоопту аныk сан түрүндө алабыз) (a+b)*c/(2*d)
$\frac{x^3-y^2}{x+y}$	(x*x*x-y*y)/(x+y)

МАТЕМАТИКАЛЫК ФУНКЦИЯЛАР

Ар түрдүү маселелерди чыгарууда биз квадраттык тамырларды, тригонометриялык функциялардын маанилерин ж.б.у.с чыгаруу зарылчылыгы менен көп кездешебиз. Мында, буларды да програмалоого туура келет. Ошондуктан бул иштерди жеңилдетүү максатында, программалоо тили менен бирге көп кездешүүчү стандарттык функциялардын программасы жазылган атайын библиотекасы да берилет.

Колдонуу үчүн, алардын жазылышын жана кайрылуу эрежелерин сактоо зарыл. Төмөнкү таблицада алардын кээ бирлери жазылды жана бул таблица тилди үйрөнүү деңгээлиңиз менен толукталып турарын эстен чыгарбоо керек.

Функция	Паскаль тилинде	Аргумент-тин типи	Жыйынтык-тын типи
$\|x\|$	abc(x)	integer real	integer real
x^2	sqr(x)	integer real	integer real
$\sqrt{x}$	sqrt(x)	integer real	real
sinx	sin(x)	integer real	real
cosx	cos(x)	integer real	real
e^x	exp(x)	real	real
аргументтин бөлчөк бөлүгүн бөлүп алуу	Frac(x)	real	real
аргументтин бүтүн болүгүн бөлүп алуу	int(x)	real	real
lnx	ln(x)	real	real
жуп, тактыгын аныктоо	odd(x)	longint	boolean (true, эгер сан так болсо)
кокус санды түзүү	random(x)	word	integer
	random;	–	real
анык санды бүтүнгө чейин тегеректөө	round(x)	real	integer longint
анык сандын бөлчөк бөлүгүн бөлүп алуу	trunc(x)	real	integer real

2. ОПЕРАТОРЛОР ЖЕ PASCAL ТИЛИНИН СҮЙЛӨМДӨРҮ

МЕНЧИКТӨӨ ОПЕРАТОРУ

Бул бардык программалоо тилдериндеги негизги операторлордун бири. Менчиктөө оператору менен өткөн параграфта таанышып кеткенибизге карабастан, дагы бир жолу кайталап өтөлү: бара-бар белгисинин сол жагында жайгашкан өзгөрмө кандайдыр бир түшүнүк тарабынан берилген жаңы маанини өзүнө менчиктейт.

Мисал: `a:=3` – бул жазуу 3 саны `a` өзгөрмөсүнө менчиктелет дегенди түшүндүрөт. `a:=a+1`; ал эми бул жазуу `a` өзгөрмөсүнүн баштапкы маанисине 1 санын кошуу менен, ошол эле өзгөрмөгө жоопту менчиктөөнү түшүндүрөт. Эгер `a:=3` болсо, анда экинчи менчиктөө оператору аткарылгандан кийин, `a` өзгөрмөсүнүн мааниси 4 болуп калат. Баштапкы мааниси сакталбайт.

Мисалы: $\frac{a^2+b}{\cos(a+b)+1{,}5}$ туюнтмасынын маанисин эсепте.

Менчиктөө операторунун жардамы менен a жана b өзгөрмөлөрүнө маани берип, туюнтманын маанисин эсептейбиз жана алынган маанини c өзгөрмөсүнө менчиктейбиз.

Программанын фрагменти:

```
a:=3.5; b:=-0.9;
c:=(a*a+b)/(cos(a+b)+1.5);
```

Операторлук кашаа. Мындан кийинки оператор жөнүндө сөз кылаардан мурун, "операторлук кашаа" деген түшүнүк менен таанышып өтөлү. Паскаль тилинде, операторлук кашаа түшүнүгүндө `Begin` (ачык кашаа) жана `End` (жабык кашаа) кызматчы сөздөрү бар.

МААЛЫМАТТАРДЫ БЕРҮҮ ЖАНА ЧЫГАРУУ ОПЕРАТОРЛОРУ

Маалыматтарды клавиатурадан окуу менен биз `read`(же `readln`) процедурасын колдонобуз. Ошондой эле бул операторду файлдын компоненттерин окууда да колдонсок болот.

Бул оператор жөнүндө маалыматтарды жалпы карайлы:

Клавиатура менен киргизүүгө кайрылуу формасы:

```
read(<киргизилүүчү элементтердин тизмеси>);
```

Текстик файл менен киргизүүгө кайрылуу формасы:

```
read(f,<киргизилүүчү элементтердин тизмеси>);
```

мында f – файлдык өзгөрмөнүн аты.

Киргизилүүчү элементтердин тизмесине char, string типтериндеги, каалаган бүтүн же анык типтеги бир же бир нече өзгөрмөлөрдүн удаалаштыгын коюшок болот. Элементтердин тизмесинде төмөнкү типтеги өзгөрмөлөр болушу мүмкүн:

- сандык(эгер ондук чекит түрүндө көрсөтүлсө, анда жалгыз гана real)

```
var a,b,c,d:real;
...
read(f,a,b,c,d);
```

- саптар(string), бул учурда машинанын эсинде өзгөрмөлөр string тиби катарында берилет. Ошондуктан саптар үчүн гана болгон операцияларды жасай алабыз. Клавиатурадан киргизилүүчү символдордун максималдуу узундугу 127 ге барабар.

write оператору информацияны тексттик файлга же экранга чыгарууну камсыз кылат. Кайрылуу форматы төмөнкүдөй:

- информацияны тексттик файлга чыгарууда:

```
write(<файлдык өзгөрмөнүн ысмы>, <чыгарылуучу
элементтердин тизмеси>);
```

- экранга чыгарууда:

```
write(<чыгарылуучу элементтердин тизмеси>);
```

writeln оператору. Кайрылуу форматы төмөнкүдөй:

- информацияны тексттик файлга чыгарууда:

```
writeln(<файлдык өзгөрмөнүн ысмы>,<чыгарылуучу
элементтердин тизмеси>);
```

Элементтердин тизмесин көрсөтпөй койсок да болот. Бул учурда саптын аяктоо белгиси eoln берилет.

```
writeln(<файлдык өзгөрмөнүн ысмы>);
```

- экранга чыгарууда:

```
writeln(<чыгарылуучу элементтердин тизмеси>);
```

Эгерде чыгарылуучу элементтердин тизмеси берилбесе, анда курсорду кийинки сапка которуу ишке ашырылат. Ошондой эле бул чыгаруу операторун жазууну ар кандай форматтары бар. Эгер берилген a өзгөрмөсүнүн тиби real – анык сан болсо, анда чыгаруу операторун writeln('a=',a:10:4); форматында жазсак бо-лот. Бул операторду аткарылышы менен, мисалы a=3456,789 болсо, анда экранда a= 3456.789 деген көрүнүш оруп алат. Демек (') – аппостроф белгисинин ичиндеги маалымат экранга өзгөртүүсүз чыгарылат, ал эми a өзгөрмөсүнүн маанисинен кийин коюшок чекит менен бөлүнүп берилген сандар, ал өзгөрмө үчүн саптан канча орун бөлүнгөнүн көрсөтөт. Биздин учурда a өзгөрмөсүнүн мааниси үчүн 10 орун бөлүнүп, анын төртөө сандын бөлчөк бөлүгү үчүн берилген. Операторду оруп

форматтагы жазылышы тиби анык болгон өзгөрмөлөр үчүн гана орун алаарын эске сала кетмекчибиз. Эгерде өзгөрмөнүн тиби бүтүн – intrger болсо, анда мындай форматты компьютер кабыл албайт, ката деп эсептейт. Андай болгон учурда writeln('b=',b:10); деп жазып койсок жетиштүү болот.

Мисалы:

```
program esep; var a,b,c:real;
begin
writeln('a,b нын маанисин киргиз');read(a,b);
c:=(a-b)/(a+b);
writeln('a=',a:8:3,'b=',b:8:3,'c=',c:8:3);
end.
```

КУРАМА ОПЕРАТОРЛОР.

Белгилеп кетүүчү нерсе, курама операторлор структуралык программалоонун жаңы технологиясында программа жазууга мүмкүнчүлүк бере турган тилдеги маанилүү курал болуп саналат.

Курама командалардын составына кирген операторлордун мүнөзүндө чектөөлөр болбойт. Алардын арасында Паскаль тилинин башка курама операторлору болушу мүмкүн. Иш жүзүндө операторлордун бардык бөлүктөрү бир бүтүн курама операторду түзөт.

```
Begin {Программанын башы}
    Begin
        Begin
            Begin
                ...
                ...
            End;
        End;
    End;
End. {Программанын аягы}
```

Сиздер "Begin "сөзүнүн саны "End "сөзүнүн саны менен дал келе тургандыгын байкаган чыгарсыздар. Программа түзүп жатканда ката кетирбеш үчүн "Begin-End" деген удаалаш сөздү бир позициядан жазып керек. Мындай удаалаш сөздөр көп болуп кеткенде, жаңылып калбаш үчүн ар бирине номер коюп жазуу өтө ыңгайлуу.

Мисалы:

```
Begin {Программанын башы}
    Begin {1}
        Begin {2}
            ...........
        End; {2}
    End; {1}
End. {Программанын аягы}
```

ШАРТТУУ ОПЕРАТОР

Тармактанган эсептөө процессинде(тармактуу алгоритм) эсептөөнүн өзүнчө этабы ар дайым эле бир түрдө чыгарыла бербестен, кээ бир шарттарга ылайык аларды чыгаруу үчүн колдонулат.

Мисалга бизге квадраттык теңдеменин маанисин табуу берилсин:

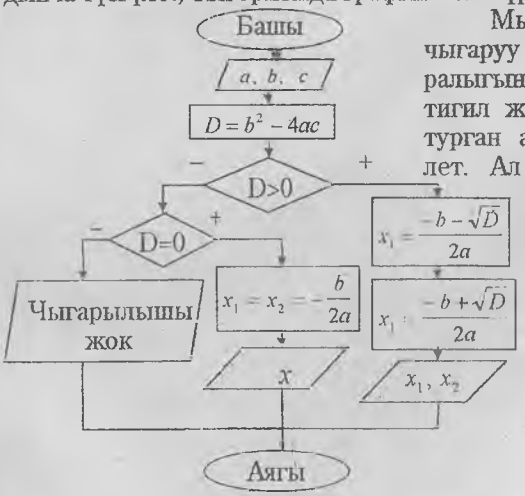
$$ax^2 + bx + c = 0$$

Бул теңдеменин маанисин алыш үчүн, чыгаруунун этаптарын санап өтөлү.

1) a, b, c коэфиценттеринин маанилерин берүү;
2) Дискриминанттын маанисин табуу;
3) Дискриминанттын маанисин текшерүү, алынган мааниден тиги же бул шартты аткаруу.

Алгоритмдин 3-пунктуна кеңири түшүнүк берели: алынган дискриминанттын маанисине анализ жүргүзөбүз, андан кийин дискриминанттын маанисине ылайык чыгарыла турган шартты тандайбыз, мында бул тармактуу алгоритмге тиешелүү экендигин көрүүгө болот.

Келгиле бул алгоритмди графикалык түрдө түзөбүз(текстин программасын мындан кийинки параграфты окугандан кийин, өз алдынча түзгүлө.) Алгоритмди графикалык түрдө көрсөтөлү:



Мындай түрдөгү алгоритмди чыгаруу үчүн, атайын шарттын тууралыгына же каталыгына жараша тигил же бул шартты тандап ала турган атайын оператор керектелет. Ал эми бул операторду шарттуу отүү оператору деп атайбыз жана аны ар түрдүүчө: толук шарттуу жана толук эмес шарттуу түрдө жазууга болот.

1. Толук шарттуу оператор.

```
If<шарт>Then<оператор1>Else<оператор2>
```

Эгер анда антпесе

Мисалы: Берилген a жана b эки сандын чоңун тап.

```
If a>b then writeln('чоң сан a=',a)
Else writeln('чоң сан b=',b);
```

Операторлордун жана кызматчы сөздөрдүн жазылышына көңүл бургула. Ар бир Else деген сөз If тин алдында жазылып турат. Жакшы окулуш үчүн, операторду кийинки сапка жазууга болот.

Көңүл бургула! Else нин алдында үтүрлүү чекит койгонго болбойт.

2. Толук эмес шарттуу оператор. Мында Else кызматчы сөзү жазылбайт. Бул учурда оператор төмөнкүдөй көрүнүштө болот:

```
If <шарт> Then <оператор1>;
```

Бул конструкция кандай иштейт? Дагы бир жолу жазабыз:

```
If <шарт> Then <оператор1>;
<оператор2>;
<оператор3>;
.....
```

1) If оператору иштейт, б.а. шарт текшерилет.

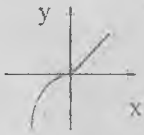
Мүмкүн болгон 2 варианты бар;

Туура шарт, бул учурда программада <оператор1> аткарылып, андан соң кийинки <оператор2>, <оператор3> аткарылат.

Жалган шарт, <оператор1> иштелбейт, ошол эле учурда ка-тар турган <оператор2>, андан кийин <оператор3> ж.б. аткарылат.

Мисалы: Аргументтин мааниси x тен көз каранды болгон $f(x)$ функциясынын маанисин тап.

$$f(x)=\begin{cases} x, \text{ эгерде } x>0 \\ 0, \text{ эгерде } x=0 \\ -x^2, \text{ эгерде } x<0 \end{cases}$$



Бул функциянын чыгарылышын төмөнкүчө жазууга болот:

```
If  x>0 then y:=x;
If x=0 then y:=0;
If x<0 then y:=-x*x;
```

Көптөгөн маселерди чыгарып олтуруп, көпчүлүк учурларда биз "then" же "else" сөзүнөн кийин операторлорду колдонуу керек экендигин түшүнөбүз. Аракеттердин тобун жазуу үчүн begin-end курама операторлорун (опертордук кашааларды) колдонобуз.

```
1. If<шарт>then Begin
                    оператор1;
                    оператор2;
                    ...
                    оператор n;
                End;
   Else оператор;
```

```
2. If <шарт> Then  оператор
   Else Begin
        оператор1;
        оператор2;
        ...
        оператор n;
        end;
3. If <шарт> Then Begin
                  оператор1;
                  оператор2;
                  ...
                  оператор n;
                   End
  Else Begin
        оператор1;
        оператор2;
        ...
        оператор n;
      End;
```

Синтаксистик түзүлүшүнө көңүл бургула: `Begin` жана `End` сөздөрүнүн ортосунда Паскаль тилинин эрежелерин сактап, б.а бири-биринен үтүрлүү чекит менен бөлүнүп бир нече операторлор турат. `Else` сөзүнүн алдында үтүрлүү чекит турбаш керек, анткени бул `If ... Then ... Else` бир бүтүн операторлор болуп саналат.

IF КЕ КАМТЫЛГАН ОПЕРАТОРЛОР

Мисал: x саны -2 ден 13 чейинки интервалда жатабы, аныктоо керек. Эң биринчиден биз эки шартты текшеребиз:
1) x (–2) ден чоң болушу керек;
2) Эгер бул шарт туура болсо, анда экинчи шартты текшеребиз: x 13 төп кичине болушу зарыл. Эгерде жогоруда айтылган шарт аткарылса, анда x саны интервалда жатат.

Бул мисалды кантип жазыш керек?

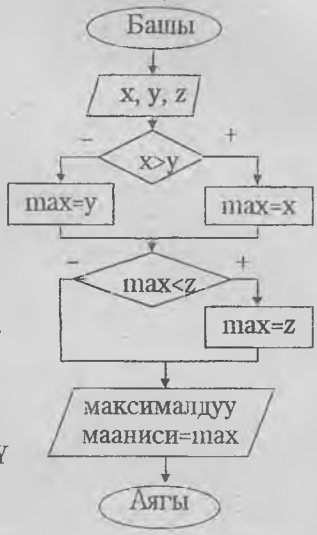
```
{биринчи шартты текшеребиз}
If x>-2 Then
{экинчи шартты текшеребиз}
If x<13 Then Writeln('x шартты канааттандырат')
Else Writeln('x>=13')
Else Writeln('x<=-2');
```

`If` ке камтылган операторлор чектелбейт. Жогоруда айтылгандар боюнча дагы бир мисал карап көрөлү.

1. Берилген 3 бүтүн сандардын эң чоң санын тап.

Мисалдын чыгарылышын структуралык схема түрүндө көрсөтөлү:

Мында x жана y өзгөрмөлөрүнүн маанилерин салыштырып, алардын ичинен чоң өзгөрмөнү тандап алып, max өзгөрмөсүнө менчиктейбиз. Андан кийин z өзгөрмөсүнүн маанисин max менен салыштырып, чоң маанисин тандап алабыз. Буларды эске алып, программасын түзсөк төмөндөгүдөй болот:




```
Program  max_xyz;
Var  x,y,z:integer;Max:integer;
Begin
Writeln('x,y,zтин маанисин бер');
Readln(x,y,z);
If x>y Then max:=x Else max:=y;
If z>max Then max:=z;
{көңүл бургула: Else кызматчы сөзү
жок}
writeln('макс. мааниси =',max);
```

2. Бүтүн x, y, z сандары берилген.

1) MAX(x+y+z, xyz) ти эсептегиле;

2) MIN(x,y,x-y) ти эсептегиле;

```
1) Program max2;
Var  x,  y,  z:integer; MAX:inteder;
Begin
Writeln('x,y,z'); Readln(x,y,z);
If x+y+z>x*y*z Then Max:=x+y+z Else Max:=x*y*z;
Writeln('чоң сан=',max);
End.
2) Program min3;
Var  x,y:integer;Min:integer;
Begin
Writeln('x,y'); Readln(x,y);
If x<y Then Min:=x Else Min:=y;
If x-y<Min Then Min:=x-y;
Writeln('эң кичине сан=',min);
End.
```

3. a, b, c анык сандары берилген. a<b<c барабарсыздыгы аткарыла тургандыгын аныкта.

```
Program n3;
Var  a,b,c:real;
Begin
Writeln('a,b,c');Readln(a,b,c);
If a<b Then If b<c Then Writeln('шарт аткарылды')
```

```
    Else Writeln('шарт аткарылган жок')
Else Writeln('Шарт аткарылган жок');
End.
```

4. Эки бүтүн сан берилген. Эгерде биринчи сан, экинчиден чоң болсо экөөнүн айырмасын тап. Ал эми кичине болсо суммасын тап.

```
Program n4
Var a,b:integer;
Begin
Writeln('a жана b өзгөрмөлөрүнө маани бергиле');
Readln(a,b);
If a>b then a:=a-b Else  a:=a+b;
Wrietln('a=',a,'b=',b);
End.
```

АРАКЕТТЕРДИН КАЙТАЛАНЫШЫ, ЦИКЛ ОПЕРАТОРЛОРУ

Цикл - программалоо тилиндеги кандайдыр бир иш аракетти аткарууда бир же бир нече командалардын кайталаныпы.

Цикл процессин уюштуруу. Кээ бир маселерге программа түзүп жатканда, берилген иш аракетти бир нече жолу аткарууга тийиш болгон учурда жогоруда айтып өткөн цикл операторун пайдалануу эң ыңгайлуу.

Мисал катарында эң жөнөкөй көбөйтүүнүн таблицасын алалы: 2 санынын 1 ден 10 чейинки сандар менен болгон көбөйтүүсүн карап көрөлү.

2 x 1 = 2	2 x 6 = 12
2 x 2 = 4	2 x 7 = 14
2 x 3 = 6	2 x 8 = 16
2 x 4 = 8	2 x 9 = 18
2 x 5 = 10	2 x 10 = 20

Таблицадан иш аракет бир нече жолу кайталанып жатканына көңүл бурабыз. Көбөйтүүнүн иш аракети жана көбөйтүнүн маанисинин табуу үчүн көбөйүүчү оңол бойдон калат жана көбөйтүүчү бирден кадамга өсүп турат. Бул мисалдын чыгарылышынын башында эле цикддын кайталануусунун саны белгилүү. Цикл менен иштөөдө өзгөрмө берилип ага баштапкы маани менчиктелет.

Паскаль тилинде кадамы бирге барабар болгон цикл процесси **параметрлүү цикл** деп аталат. Параметр циклына жогорудагы таблица мисал боло алат. 2 x i = a

i нин ордунда бирден оңго чейинки, ал эми a нын ордунда 2 ден 20 га чейинки сандар өзгөрөт. Демек, мындай жыйынтык чыгарабыз: 2 x i = a сабы 10 жолу кайталануу үчүн цикл операторун колдонобуз. Цикл операторунун синтаксистик жазылышы төмөнкүдөй;

```
FOR      i:=N        TO        k          DO
үчүн  баштапкы маани      акыркы маани   жасоо
 оператор;
```

Эгерде бир нече иш аракеттердин тобун аткарыш керек болсо, оператор төмөнкүдөй көрүнүштө болот:

```
For i:=n to k do
 Begin
    оператор1;
    оператор2;
    ...
    оператор n;
 End;
```

Көңүл бургула!

1. Цикл операторунун улам кийинки процессинде эсептегич (счетчик) бир кадамга өзгөрүп турат.
2. Оператордун конструкциясынын туура жазылышына төмөнкүнү эстеп калуу керек: i, n, k өзгөрмөлөрү "иреттелген тип" гана болоалышат, ар бир маалыматтын өзүнүн номери, башка маалыматтын арасындагы өзүнүн белгилүү орду болуш керек. Бул жөнүндө биз калган параграфтарда кеңири түшүнүктөрдү алабыз. (биз азырынча иреттелген тип – integer жөнүндө түшүнүк алдык.)
3. Эсептегич i ге баштапкы гана маани менчиктелерин унутпоо керек.
4. Эгерде баштапкы маани акыркы маани менен дал келсе, цикл операторлору бир гана жолу аткарылат.
5. Эгерде баштапкы маани акыркы маанидеи чоң болсо, анда цикл операторубир да жолу аткарылбайт.
6. Циклден чыгууда эсептегичтин(счетчиктин) мааниси акыры маани менен дал келет: i:=k

Мисалы: 10 дон 20 га чейинки натуралдык сандардын квадратын экранга чыгар.


```
Program n1;
Var i:integer;{циклдин счетчиги}
    A:integer;
Begin
for i:=10 to 20 do
 Begin
 A:=i*i;
 Writeln('сандын квадраты',i,'=',A);
 End;
End.
```

же

```
For i:=10 to 20 do
Writeln('сандын квадраты',i,'=',A);
End.
```

Бир кадамга кичирейгендеги "FOR" циклинин конструкциясын карап көрөлү. Оператор төмөнкүдөй формада жазылат:

```
For      i:=n      downto      k              do
        баштапкы маани   төмөн   акыркы маани
  оператор1;
  {же begin жана end операторлук кашааларына кирген
  операторлордун тобу}
```

Мисалы: экранга 40 тан 10 го чейинки натуралдык сандарды чыгар.

```
Program n2;
Var i: integer;
Begin
For i:=40 downto 10 do Write(i,'');
End.
```

БАЗАЛЫК АЛГОРИТМДЕР

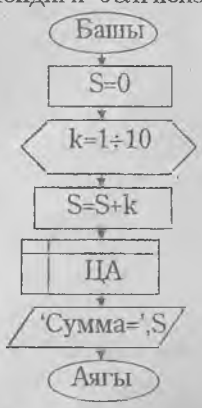
Бул бөлүмдү өтө көңүл буруп окушуңар керек. Биз баардык маселе мисалдарда иш жүзүндө колдонуучу алгоритмдерди карап чыгабыз. Мындай алгоритмдер көп деле эмес, аларды **базалык алгоритм** деп аташат. Аларды жакшы түшүнүп, эсиңерге түйүп калууга аракеттенгиле.

n сандын суммасын эсептеп, чыгаруучу алгоритм.

Келгиле n натуралдык сандарын алып төмөнкү мисалды карап көрөлү: n=1 ден 10 го чейинки натуралдык сандардын суммасын табуу.

$$S = 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10$$

Бул мисалдын маанисин Паскалда чыгаруу үчүн аны формалдаштырып көрөлү. Мында ар бир кошуучунун эмнеге барабар экендиги белгисиз болгондуктан, бир кошуучуну k тамгасы менен белгилеп ал эми суммасын S менен белгилейли. Кошуу процессинин саны кошучулардын санына дал келет. Бизде 10 кошулуучу болгондуктан 10 жолу кайталоо үчүн цикл уюштурушубуз керек.



Мындай суроо коюлушу мүмкүн: биринчи кошуучу жана экинчи кошуучу кайсы? k тамгасы менен 1 ден 10 го чейин өзгөргөн ар бир кийинки кошулуучуну белгилеп, жыйынтыгын S озгөрмөсүнө менчиктейбиз: S := ? + a;

сумма кошуучу кошулуучу

Бул менчиктөө командасы 10 жолу кайталанууга тийиш. Ал эми кошуучуну ордупа S өзгөрмөсү турат. Анын баштапкы мааниси дайыма нолго барабар. Цикл операторунун биринчи процессинде S өзгөрмөсүнө k = 1 кошулуп биринчи сумма S = 1 ди алабыз. Ал эми экинчисинде S = 1 ге k = 2 ни ко

шуп, S = 3 суммага ээ болобуз. Сумманын мааниси уламдан улам өсүп отурат. Андан ары кайталаныш отуруп, аягында k = 10 ду коюп, S = 55 маанисин алабыз. Алгоритмдин график түрүндө чагылдырылышы оң жактагы сүрөттө көрсөтүлгөн. Ал эми программасы төмөнкү көрүнүштө болот:

```
program summa;
var  i,s:integer;
begin s:=0;
     for k:=1 to 10 do s:=s+k;
     writeln('summa=',s);
end.
```

Бул процесстин машинада кандай ишке ашырыла турганын байкап көрөлү:

1 цикл	2 цикл	3 цикл	X цикл	
k:=1; S:=0;	k:=2; S:=1;	k:=3: S:=3;	k:=10, S:=45;	Циклдан
S:=S+k;(1=0+1)	S:=S+k;(3=1+2)	S:=S+k;(6=3+3)	S:=S+k;(55=45+10)	чыгуу
k:=1+1=2;	k:=2+1=3;	k:=3+1=4	k=10+2	k:=10;
k<=10? Ооба	k<=10? Ооба	k<=10? Ооба (ж.б.)	k<=10? Жок	S:=55;

1-Маселе: Берилген 12 бүтүн сандын суммасын тап.

```
Program;
Var a:integer; {киргизилген сан}
i:integer; {сандын счетчиги}
S:integer; {сумма}
Begin
S:=0;
For i:=1 to 12 do Begin
 Writeln('кийинки кошулуучуга маани бер');Readln(a);
 S:=S+a; End;Writeln('summa=',s);
End.
```

2-Маселе. 50 дөн 600 гө чейинки натуралдык сандардын суммасын тап.

```
Program;
Var i:integer; {сандын счетчиги}
S:integer; {сумма}
Begin S:=0;
For i:=50 to 600 do S:=S+i;
Writeln('summa=',s);
End.
```

Берилген шартты канаатандыруучу сандардын эсебин аныктоо.

Бул алгоритмди конкреттүү мисалдарда карап көрөлү, клавиатурадан киргизилген 30 сандын арасынан терс сандардын санын аныктагыла. Эгерде биз мындай сандардын эсебин билгибиз келип жатса, биз 0 го барабар болгон жана улам бир кадамга өскөн өзгөрмөнү киргизишибиз керек. Кайсы оператор буга жардам бере алат? Албетте,

шарттуу өтүү оператору. Эгерде силер биринчи алгоритмди жакшы түшүнсөңөр, бул алгоритм эч кандай кыйындык деле алып келбейт.

Алгоритмди графика түрүндө көрсөтөлүшү оң жакта берилген.

Ал эми программасы мындай болот:

```
Program n;
Var a,k,i: integer;
begin k:= 0;
For i:=1 to 10 do begin
write('a нын маанисин бер');
readln(a);If a<0 then k:=k+1;
end;
If k=0 Then writeln('терс
сандар жок')
Else Writeln('терс сандардын
саны',k);
end.
```

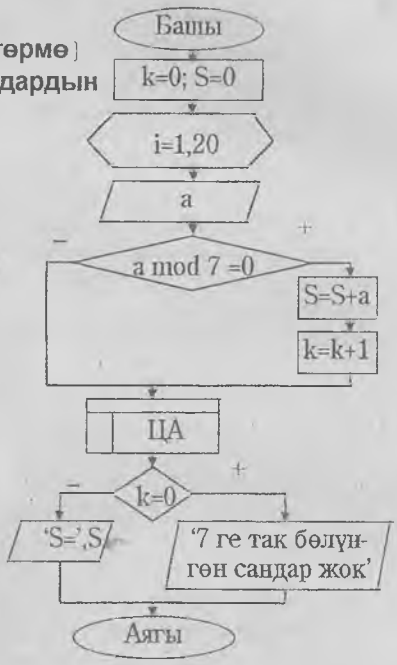
Жогорудагы эки алгоритм чогуу катышкан мисалды иштеп көрөлү.

Клавиатурадан киргизилген 20 сандын ичинен 7 ге так бөлүнгөн сандардын суммасын эсепте.

Бул мисалды иштөө үчүн сумманы табуунун алгоритми өтө керек экендиги талашсыз. Бул типтеги мисалдарды чыгаруудагы негизги каталардын бири – берилген шартты канаатандыруучу сандардын жоктугу эсептелет. Шарт боюнча эгерде сумма 0 гө барабар болсо, анда 7 ге так бөлүнгөн сандар жок болуп саналат. Бирок мындай деп айтууга да болбойт. Биз клавиатурадан -21 жана +21 маанилерин киргизип көрсөк, алардын суммасы 0 гө барабар болот. Мындай кырдаалды анализдөөнүн бирден бир варианты болуп 7 ге так бөлүнгөндөрүнүн санын аныктоо, андан кийин эсептегичтин маанисин анализдөө болуп саналат. Эгерде эсептегич 0 гө барабар болсо, анда сумма эсептелбейт. Белгилей кетчү нерсе: бул бирден-бир гана вариантардан болуп эсептелет.

Бул берилген алгоритмдин блок схемасы жана программасы төмөнкүдөй болот:

```
Program n2;
Var a:integer;{киргизилүүчү өзгөрмө}
k:integer; {7ге так бөлүнгөн сандардын
           эсептегичи}
s:integer;{сумма}
i:integer;{циклдин счетчиги}
begin k:=0;s:=0;
for i:=1 to 20 do  begin
 write('a нын маанисин бер');
 readln(a);
 if a mod 7=0 then begin
 k:=k+1;s:=s+a;end;end;
if k=0 Then writeln('7ге так
бөлүнгөн сандар жок');
else writeln('7ге бөлүнгөн
сандардын суммасы=' ,s');
end.
```



n! ди эсептөөнүн алгоритми

n! дегенибиз төмөнкү көбөйтүндүгө барабар чоңдук: n! = 1 • 2 • 3 • 4•... •n. Мисалы 5! ды эсептесек, анда 5! = 1•2•3•4•5 = 120. Ал эми 0! = 1 деп кабыл алынган.

Эсептөө алгоритми иш жүзүндө сумманы эсептөөнүн алгоритминен эч айырмаланбайт, кошуунун иш аракетин көбөйтүүгө алмаштырып, ал эми көбөйтүндү чогулуучу баштапкы өзгөрмөнүн маанисин 1 ге барабар деп карайбыз.

Программанын тексти төмөнкү түрдө болот:

```
Program n;
Var f:integer;i:integer;
Begin f:=1;
For i:=1 to 5 do f:=f*i;
Writeln('5! =',f);
End.
```

a^n ди эсептөөнүн алгоритми.

n натуралдык сан болгон, a^n эсептөө алгоритмин төмөнкүчө карасак болот: $\underbrace{a^n = a \bullet a \bullet ... \bullet a}_{n\ жолу}$

Бул бизге бирдей болгон көбөйтүүчүлөрдү көбөйтүүдөн келип чыккан кобойтүндүнү берет. Мында биз керектүү коррективдерди киргизип, жогоруда колдонулган алгоритмди пайдаланабыз:

```
Program n3;
Var a:integer;{негизи}
    n:integer;{даражанын көрсөткүчү}
    w:integer;{жыйынтык}
    i:integer;{циклдин счетчиги}
begin write('a негизинин маанисин бер');read(a);
write('n дин көрсөткүчүн киргиз');read(n);
w:=1; for i:=1 to n do
w:=w*a; writeln(a'нын'n,'даражасы=',w,);
end.
```

Мисалдар.

1. n = 10 болгондо 1!+2!+3!+...+n! ди эсептөө.

```
...
s:=0;{сумманы чогултуучу өзгөрмө}
f:=1;{n! ди чогултуучу өзгөрмө }
for i:=1 to 10 do{эки базалык алгоритмди пайдаланабыз}
                 {n! ди эсептөө жана сумманы эсептөө}
begin f:=f*i; s:=s+f; end;
```

2. Эсепте: a(a-1)(a-2)...(a-(n-1)), a - аныk сан, n – натуралдык сан.

```
...
Readln(a,n);P:=1;
For i:=1 to n do P:=p*(a-(i-1));
```

АЗЫРЫНЧА ЦИКЛИНИН КОНСТРУКЦИЯСЫ

Параметри менен берилген циклдин конструкциясын жетишерлик деңгээлде карап чыктык. Эң негизги эске тутуп калчу нерсе: циклдин канча жолу кайталанышы анын аткарылышынын башында берилери болуп саналат. Бирок кээ бир учурда кайталоонун саны башында берилбей цикл процессин иштеп жаткан учурда берилиши мүмкүн.

Мындай мисалды карап чыгалы: эгерде суммалоо 3 төн башталса жана ар бир кошулуучу мурдагыга караганда 2 ге чоң болсо, суммасы 200 дөн ашкан натуралдык сандардын санын аныктагыла.

Бул мисал жогоруда биз карап чыккан мисалдан эмнеси менен айырмаланат? Сумманы табуу алгоритми ошол эле, бирок бизге кошулуучулардын саны белгисиз. Мындай түрдөгү мисалдарга параметри менен берилген циклдин конструкциясы жарабайт. Паскаль тилинде жогорудагыдай мисалдарды чыгаруу үчүн эки түрдөгү: азырынча жана чейин циклдери бар. Алардын ар бироону кененирээк карап чыгалы.

```
While <шарт> do
оператор{же Begin - end операторлук кашаалары
         операторлордун тобу}
```

Кыргыз тилиие бул фразаны которгондо мындай болот: “Азырынча шарт туура болуп жатса, анда цикл оператору аткарылсын”.

Буларды эске тутуп калуу өтө маанилүү:

1. Эгерде циклдин биринчи кайрылуусунда шарт жалган болсо, анда цикл иштебейт.
2. Берилген конструкция параметри менен берилген циклдин конструкциясындай иштей алат. Бирок менчиктөө операторунун жардамы менен эсептегич(счетчик) катары эсептелип жаткан өзгөрмөнү өзгөртүп туруу зарыл. While конструкциясынын жардамы менен силер өзүңөргө ылайык кадамды тандап ала аласыңар.

Жогоруда берилген мисалды карап чыгалы.

Алгоритмдин графикалык түрү:


```
Program n;
Var t,k,s:integer;
Begin s:=0;k:=3;t:=0;
While s<=200 do
 Begin
  s:=s+k;k:=k+2;t:=t+1;
 End;
Writeln('кошулуулардын саны=',t);
End.
```

Көңүл бургула!

1) Циклдан S =200 же S > 200 болгондо гана чыга аласыңар.
2) Кошулуучулардын санын эсептегич t болуп саналат. Алгачкы учурда t=0. Баштапкы кошулуучунун мааниси k = 3 жана ал улам 2 ге өзгөрүп отурат(k=k+2). k нын маанисинин өзгөрүшү менен t улам 1 ге чоңоет. Качан гана S тин мааниси 200 гө барабар же жогорулап кеткенде, кошулуучулардын санын эсептегич t нын мааниси экранга чыгат.

2-Мисал. Биринчи 50 натуралдык сандын суммасын эсептеп чыгаруу.

Бул мисалды For оператору менен чыгарганбыз. Ал эми бул жолу чыгаруунун экинчи варианты While дын конструкциясы менен кандай иштерин карап көрөбүз.

```
Program n;
Var i:integer;{кезектеги кошулуучу - эсептегич}
    s:integer;{сумма}
Begin i:=1; s:=0;
While i<=50 do
  Begin s:=s+i;i:=i+1; end;
Writeln('Сумма =', s)
End.
```

3-Мисал. 20 дан 70 ке чейинки жуп сандардын суммасын эсепте.

```
Program n
Var k:integer; s:integer;
```

```
Begin k:=20; S:=0;
While k<=70 do
 Begin s:=s+k; k:=k+2; End;
Writeln('20 дан 70 ке чейинки жуп сандардын
суммасы=',s);
End.
```

ЧЕЙИН ЦИКЛИНИН КОНСТРУКЦИЯСЫ

Чейин циклинин конструкциясы төмөнкүдөй болот:

```
Repeat
.....
Until <шарт>;
```

Паскаль тилинин атайын кызматчы сөзү Repeat (англисче - кайталоо) циклдин операторлорун ачат, андан кийинки операторлор аткарылып, циклдан чыгуу үчүн берилген шарт текшерилет. Конструкциясы төмөнкүчө которулат: "Шарт туура болгончо кайтала"

Көңүл бургула!

1. Цикл жок дегенде бир жолу аткарылат.
2. Циклда иштеп жатып, биз шарттын тууралыгына келишибиз керек.

Кийинки сандын квадраты 625 тен чоң болгон убакка чейинки натуралдык сандардын квадраттарынын суммасын эсептеп чыгалы.

```
Program n;
Var  k:integer;{кошулуучулар}
     s: integer;{сумма}
begin s:=0;k:=1;
repeat
s:=s+k*k; k:=k+1;
until k*k>625;
writeln('s=',s);
end.
```

Программа түзүп жатканда ар бир берилген маселе-мисалдардын үстүнөн абдан ой жүгүртүп, кайсы конструкция оптималдуу болсо ошону менен иш жүргүзүү керек.

Мисалы: клавиатурадан киргизилген бүтүн сандардын арасынан биринчи терс санды жолуктурганга чейинки сандардын суммасын эсептейли. Келгиле ойлонуп көрүп, циклдин кайсы оптималдуу конструкциясы бул маселени чыгарууга керек экендигин табалы.

1. For операторунун конструкциясы жарабайт, анткени цикл оператору канча жолу кайталанылат экендигин билбейбиз.
2. Ал эми азырынча циклинин Repeat-Untile конструкциясын колдонууга болот. Мисалда түшүндүрүп көрөлү.

```
Var  a,s:integer;
```

```
Begin s:=0;
a:=0;{биз бул жасалма кадамды жасоого мажбур болобуз}
Repeat
S:=s+a; Readln(a);
Until a<0;
  ....
```

Киргизүүнүн иш аракеттери суммалоодон кийин турсун үчүн, а өзгөрмөсүн 0 гө барабарлашыбыз керек болду, суммалоо операциянын биринчи аткарылышы а өзгөрмөсүнүн 0 гө барабар маанисинде иштейт жана албетте бул биз мажбур болгон жасалма кадам болуп эсептелет.

Берилген иш аракетти колдонбой турганда эмне болоорун карап көрөлү.

```
S:=0;
Repeat
readln(a);S:=s+a;
Until a<0;
.....
```

а = -10 ду киргизебиз. Цикл аткарыла баштайт, андан кийин циклдан чыга баштаганда мисалдын шарты туура келбеген -10 го барабар болгон сумманы алабыз. Себеби биз суммадан терс кошулуучуну алып салышыбыз керек болучу.

3. Азырынча циклинин конструкциясы While менен маселени чыгаралы.

Бул конструкцияны колдонууга болот, бирок циклга биз шарттын тууралыгында гана кире алабыз. Циклга кирүү үчүн а өзгөрмөсүнүн маанисин циклга чейин киргизишибиз керек.

```
Var a,s:integer;
Begin readln(a);s:=0;
while a>0 do
 begin s:=s+a;readln(a);end;
writeln('сумма=',s)
End.
```

Ошентип биз мындай жыйынтыкка келдик: мындай типтеги маселе-мисалдарды иштеш үчүн азырынча же чейин циклинин конструкцияларын колдонууга болот. Экөөнүн кайсынысын колдонууну өзүңөр тандайсыңар. Эң негизгиси тандап алган конструкцияларды так, катасыз аткарууга аракеттенгиле.

Көңүл бургула! Төмөнкү маселелерди чыгарып жатканда алынган элементтер сакталбайт.

Дагы бир мисал карап көрөлү.

Бүтүн сандар берилген(бардыгы 30). 3 кө так бөлүнгөн сандардын суммасын эсептегиле.

Бул сумманы эсептөөдөгү колдонулуучу базалык алгоритм. Бирок бул алгоритмди биз шартты канаатандыруучу сандар үчүн колдонобуз, албетте, андан тышкары шарттын чындыгын же жалгандыгын текшеребиз, шартты канааттандыруучу элементтердин жок болгондогу абалын анализдейбиз.

```
Program n1;
Var  a:shortint; i:byte; k:byte; s:integer;
begin s:=0; k:=0;
for i:=1 to 30 do begin
  a=random(50)+2;
  if a mod 3 = 0 then begin k:=k+1; s:=s+k; end;
  end;
if k=0 then writeln('шартты аткаруучу сан жок')
else writeln('3кө так бөлүнгөн сандардын суммасы,
S=',s);
end.
```

Дагы бир жолу маалыматтардын тибине кайрылалы. Кайрадан базалык алгоритм – n! га кайрылып n дин мааниси: 5, 6, 7, 8 болгондо карап чыгалы.

(f = n! - integer, n - integer)

N = 5 те	N = 6 да	N = 7 де	N = 8 де
f = 120	f = 720	f = 5040	f = -25216

Мында биз көрүп тургандай n = 7 ден кийин, программа түшүнүксүз жыйынтык берип жатканын көрүп турабыз. Келгиле, n өзгөрмөсүнүн тибин жаңа жыйынтыктын тибин аныктайбыз. Экөөнү тең башында integer деп жарыялаганбыз. Берилген n үчүн n! ды калькулятордон эсептеп көрөлү.

N = 5	N = 6	N = 7	N = 8
N! = 120	N! = 720	N! = 5040	N! = 40320

8! ды калькулятордон эсептеп 40320 деген көп орундуу санды алдык. Мындан биз чектелген санды аттап кеткенде туура эмес жыйынтыкка ээ боло баштайбыз. Демек типти туура эмес колдонуу, туура эмес жыйынтыкты көрсөтөт. Типтерди жакшылап өздөштүрүш үчүн, кийинки параграфты эң жакшы өздөштүрүшүбүз керек.

CHAR МААЛЫМАТТАР ТИБИ

Паскаль программалоо тили символдук(литердик) чоңдуктар менен иштөөгө да мүмкүнчүлүк берет. Символдук маалыматтар же туруктуу (константа), же өзгөрмө болушу мүмкүн.

Мындай маалыматтарды иштетүү үчүн char тибин колдонулат. Char тибинин маанилери болуп дисплейдин экранынан көрүнгөн бардык символдор: сандар, тамгалар, катыш белгилери

атайын сим-волдор ж.б. эсептелинишет. Char тибинде баяндалган өзгөрмө бир гана символдун маанисин ала алат. Бул маанини өзгөрмөгө ыйгаруу операторунун жардамы менен же клавиатурадан киргизип ыйгарсак болот.

1-Мисал

```
var lim:char;
...
lim:='a';
```

Ыйгарылуучу символ милдеттүү түрдө апострофтордун ортосунда жайгашышы керек.

Көңүл бургула! Качан гана char тибинин мааниси программада көрсөтүлсө, анда милдеттүү түрдө символдун эки жагына тең апостроф белгиси коюлушу керек. Литердик типтеги өзгөрмөнүн маанисин чыгарууда апострофтор чыгарылбайт.

Эгерде write(lim) операциясын иштетсек, анда экранда "a" тамгасы пайда болот.

Тилдин символдору бир тартипте болгондуктан, символдук маалыматтарга катыш белгилерин: <, <=, <>, >, >= колдонуугa болот. Ошондуктан 'A' < 'B', '+' < '-', ж.б. Салыштыруу операцияларынын жыйынтыгы болуп логикалык константа TRUE(чын) же FALSE (жалган) эсептелет.

Символдук маалыматтар катарында түзүлгөн функциялар колдонулат. Аларды бир тартиптүү типтер менен болгон иштерге да колдонсок болот.

Багытталышы	Функция-нын ысмы	Аргумент-тин тиби	Жыйынтыктын тиби
Катар номерди калыбына келтирүү	Ord(x)	бүтүн	өзүнүн мааниси
Ord(x)-1 ге туура келген маанини калыбына келтирет	Pread(x)	символдук	0 дөн 255 ке чейинки диапазондогу оң сан
Ord(x)+1 ге туура келген маанини калыбына келтирет	Succ(x)	символдук	0 дөн 255 ке чейинки диапазондогу оң сан
byte тибиндеги түшүндүрмөнү символго жана бул символду өзүнүн мааниси катарында кайра түзөт	Chr(x)	byte	char

1-мисал. Клавиатурадан удаалаш 15 символ киргизилет. Киргизилген символдордун арасынан 'c' тамгаларынын санын эсептегиле.

```
program P_N;
```

```
var a:char; {киргизилүүчү өзгөрмө}
    k:byte;{тамгаларды эсептегич}
    i:byte;{циклдин эсептегичи}
begin k:=0;
for i:=1 to 15 do begin readln(a);
if a='c' then k:=k+1; end;
if k=0 then writeln('c тамгалары жок')
else writeln('c тамгаларынын саны=',k);
end.
```

2-мисал. '!' символу жолукканга чейин клавиатурадан удаалаш түрдө символдор киргизилет. ';' символдорунун санын эсептегиле. Маселенин шарты боюнча киргизилүүчү символдордун саны белгисиз, ошондуктан биз while же repeat циклинин конструкциясын колдонобуз.

```
Program p_n;
var lit:char; {киргизилүүчү өзгөрмө}
    k:byte; {';' белгисин эсептегич}
begin
{бул учурда киргизүү операциясын циклга чейин берүү
зарыл}
readln(lit);
while lit <>'!' do
 begin if lit=';' then k:=k+1; readln(lit); end;
if k=0 then writeln(';белгиси жок')
else writeln('; белгисинин саны=',k);
end.
```

BOOLEAN МААЛЫМАТТАР ТИБИ

Стандарттык boolean тиби true жана false түшүнүктөрүн гана ала турган каалаган элементти түшүндүрө турган маалыматтардын тиби болуп саналат. Бул чондуктар төмөнкү ыкма менен ишке ашат: False<True. Маалыматтардын тиби boolean ге 1 байт талап кылынат. Биз туура маанини 1 деп, ал эми жалган маанини 0 деп эсептесек болот.

Логикалык константаларды (True, False), өзгөрмөлөрдү жана стандарттык өзгөрмөлөрдү өзүнө камтыган маалыматтардын логикалык тиби логикалык шарттарды көчүрүүгө жардам берет. Логикалык өзгөрмөлөрдүн көнчүлүк бөлүгү программалардын операторлорунун аткарылышын башкаруу үчүн пайдаланылат. Мындай иш-аракеттерди аткаруу үчүн байланыш жана салыштыруу операцияларын пайдаланууга болот.

Байланыш операциясынын жардамы менен мамиленин түшүндүрмөсүн жасоого болот, мисалы:

$$\underbrace{A*B}_{\text{түшүндүрмө}} \quad \underbrace{\geq}_{\text{байланыш}} \quad \underbrace{C-D}_{\text{түшүндүрмө}}$$

A>0; x ≥y ж.б

Түшүндүрмөлөрдү иштетип жатканда эң биринчи сол жактагы жана оң жактагы түшүндүрмөлөр өзгөрмөлөрдүн конкреттүү мааниси үчүн эсептелет, андан кийин байланыш операциясынын тууралыгы текшерилет. Эгерде байланыш канааттандырса, анда мамилелердин түшүндүрмөсү төмөнкү маанини алат: "чын"(true), ал эми тескери учурда "жалган" (false) деген маанини алат.

Байланыштарды түшүндүрүүнүн жыйынтыгын менчиктөө операторунун жардамы менен же логикалык өзгөрмөгө менчиктесе болот.

1-Мисал: Берилген эки сандын чоңун экранга чыгар.

```
Var a,b:real; L:boolean;
Begin Readln(a,b); L:=a>b;
If L then Writeln('эң чоң сан a=',a)
Else Writeln('эң чоң сан b=',b);
End.
```

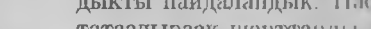
Берилген мисалда L логикалык өзгөрмөгө a>b нын байланышынын түшүндүрмөсүнүн жыйынтыгы менчиктелет. Логикалык өзгөрмө L дин мааниси a жана b сандык өзгөрмөлөрдүн маанисинен көз каранды.

КУРАМА ШАРТТАР

Маселе: x чекити (-5,12) интервалында жатаарын аныкта. Бул маселени чыгарыш үчүн if операторун пайдаланса болот.

```
If x>-5 then If x< 15 then
Writeln('чекит интервалда жатат');
```

x санынын интервалга тиешелүү экендигинин шарты: "Эгерде x тин мааниси -5 тен чоң жана 15 тен кичине болсо, анда x саны интервалда жатат." Байланыштын эки операциясын биз "жана" сөзү менен байланыштырдык, графикалык түрдө интервалга тиешелүүлүгүн төмөнкүдөй сызма менен көрсөтсөк болот. Графикалык түрү төмөнкүчө болот:



Шарттуу өтүү операторунда шарт катарында барабардыкты же барабарсыздыкты пайдаландык. Паскалда бир нече жөнөкөй шарттардан турган татаалыраак шарттарды түзүүгө болот. Ар бир жөнөкөй шарт кашааларга камтылат.

Жөнөкөй шарттар AND, OR жана NOT кызматчы сөздөрү менен жана XOR операциясы менен бириктирилет. AND, OR, NOT, XOR кызматчы сөздөрү менен берилген шарттарда логикалык операцияларды жүргүзөбүз. Курама шарттар жөнөкөй шарттар сыяктуу эле шарттуу, ажырыпча жана чейин операторлорунда пайдаланылат.

СТРУКТУРАЛЫК ТИПТЕР. МАССИВДЕР

Структура(лат. structura) – объектинин бүтүндүгүн жана бөлүгүн камсыз кылуучу туруктуу байланыштардын көптүгү, б.а. ар кандай ички жана сырткы өзгөртүүлөрдөгү объектинин негизги касиеттеринин сакталышы, курулушу, жайгашышы, тартиби.

Төмөнкү маселени карап чыгалы: клавиатурадан 15 анык санды киргизип жана алардын суммасын эсептейли, ошол эле учурда ар бир санды кийинки эсептөөлөр үчүн эске сактоо керек болсун. Бизге клавиатурадан 15 санды киргизишибиз керек, албетте бул өтө ыңгайсыз.

Паскаль өзүнүн конструкциясы боюнча өтө оор типтер менен иштегенге мүмкүнчүлүк берет. Алар колдонуучунун маалыматтар тиби деп аталат жана структуралык типтерге тиешелүү болот. Структуралык типтин өзгөрмөлөрү бир элементтен эмес, бир нече элементтерден(бардык стандарттык типтер сыяктуу бүтүн, анык, символдук жана логикалык) турушу менен айырмаланат. Структуралык типтерге массивдер, файлдар, жазылыштар ж.б. кирет.

Массив – а) бир типтеги (array, file) маанилердин көптүгү жайгаша ала турган ЭЭМдин эсинин областы, б) бир багытта бириктирилген жана бир ысымга ээ өзгөрмөлөрдүн тобу.

Массивдин элементи - массивде камтылган өзгөрмө.

Массивдин өлчөмү – массив камтыган элементтердин саны.

Индекс – маалыматтарды бири-биринен айырмалай турган математикалык сандык же тамгалык көрсөтмө. англ. Index

Массивдин элементинин индекси массивдеги элементтин номери.

Массивдин өзгөчөлүгү төмөндөгүдө: маасивдин бардык элементтери бир типтин маалыматы болуп саналат.

Маасивдерге ысым берүүдө жөнөкөй типтердеги өзөгөрмөлөрго коюлуучу талаптар коюлат.

Массивдин элементине кирүү мүмкүнчүлүгүн төмөндөгүдөй карасак болот. Массивди шарттуу түрдө n бөлүнүүчүсү бар тик бурчтук түрүндө сүроттөөгө болот, ар бир бөлүнүүчү – өзүнүн номери (индекси) бар массивдин элементти. Индекс ар дайым массивдин жанына, чарчы кашаанынын ичине жазылат.

A[1]	-23
A[2]	12
A[3]	34
A[4]	-34
.....	.....
A[N-1]	15
A[N]	50

Мында A[1] – массивдин биринчи элементи; A[5] – маасивдин бешинчи элементи; A[i] – массивдин i-элементи, акыркы мисалда индекс каттары i өзгөрмөсүн көрсөттүк. Паскалда ар бир индексти көрсөтүүчү өзгөрмө жарыяланыш керек. Индекстин тиби катары `longint` тен тышкары бардык типтерди колдонгонго болот. Массивдин тибин `TYPE` кызматчы сөзү менен сүрөттөгө болот.

Параграфтын башында берилген мисалды карап көрөлү. Эң биринчи типтердин сүрөттөлүшүнө өтүүдөн мурда, массив тибине тиешелүү анын ысмын жана өзгөрмөнүн ысмын аныктап алалы. Мейли, `mas` – типтин ысмы, a – массив тибинин өзгөрмөсүнүн ысмы болсун. Типти сүрөттөөдө биз төмөнкүлөрдү белгилей кетишибиз керек: массив тиби жарыяланат, бул үчүн `array` кызматчы сөзү пайдаланылат, индекстин өзгөрүлүшүнүн чегин белгилөө. Бизде 15 элемент бар, индекс 1 ден 15 ке чейин өзгөрөт жана ал төмөнкүдөй жазылат 1..15, мындай жазуу – диапазон деп аталат.

Жогорудагы айтылыш кеткен сөздөр Паскаль тилинде жазылышы: `TYPE mas=array[1..15] of integer;`

```
{колдонуунун тиби жарыяланат - массив, массивдин
бардык элементтери бүтүн сандар болуп саналышат}
    VAR A:mas; {массив тибине тиешелүү өзгөрмө
                жарыяланат}
```

А өзгөрмөсүнүн элементтеринин тиби массив болоорун унутпоо керек. Массив константа болуп жарыяланышы мүмкүн. Баштапкы константанын мааниси катарында – бири-биринен үтүр менен ажыратылган массив константалардын тизмеси колдонулат; тизме тегерек кашаанын ичине жазылат.

Мисалы: `Const Mas: array[1..10] of byte = (31,30,31, 30,31,31,30,31,30,31)$;`

Эске сактап калууга керек болгон бирдей типтеги маалыматтардын саны көп болгондо, массивдер керектелет.

БИР ӨЛЧӨМДҮҮ МАССИВДЕР

Элементтин жайгашуу ордун көрсөтүү үчүн бир эле индекс жетиштүү болсо, мындай массивди бир өлчөмдүү массив деп аташат. Өзгөрмолорду баяндоо бөлүгүндө биз массивдин тибин, андан кийин өзгөрмону берилген типке менчиктегенбиз. Компьютердин эсинен массивдин элементтеринин жайгашышы үчүн орун бөлүнөт. Типти баяндоодо жана программанын иштөө процессинде элемент-

тердин санын өзгөртүүдө массивдин өлчөмдүүлүгү аныкталаарын өзгөчө белгилеп кетүү керек. Аз сандагы элементтерди да колдонууга болот.

Алгач биз массивди толтуруу менен алек болобуз, б.а. ар бир элементтин өзүнө маани беребиз. Андан кийин берилген алгоритм боюнча массивди иштетебиз. Билүүгө зарыл болгон негизги алгоритмдердин бир нече түрлөрү бар. Мындай алгоритмдер базалык болоорун мурунку темаларда карап кеткенбиз. Эмесе массивди иштетүүгө мүмкүн болгон базалык алгоритмдерди санап өтөлү:

1. Бир өлчөмдүү массивдин маанилерин берүү
2. Бир өлчөмдүү массивдин маанилерин экранга чыгаруу
3. Бир өлчөмдүү массивдин элементтеринин суммасын эсептөө
4. Берилген шартты канааттандырган бир өлчөмдүү массивдин элементтеринин санын эсептөө
5. Бир өлчөмдүү массивдин максималдык(минималдык) элементин жана анын катар номерин аныктоо

Жогоруда көрсөтүлгөн базалык алгоритмдердин ар бирине өзүнчө токтололу.

1. Бир өлчөмдүү массивдин маанилерин берүүнү
   - маанини клавиатурадан киргизүү менен;
   - кокус образ менен;
   - формула боюнча жүргүзсөк болот.

Үч учурда тең биз цикл уюштуруу менен иш алып барабыз.

Мейли биз 20 элементтен турган массивге маани берели.

- бир өлчөмдүү массивге клавиатурадан маани берели.

```
for i:=1 to 20 do begin
writeln(i,'-элементти киргизгиле');
read(b[i]); {массивдин элементтерин киргизүү}
end;
```

- бир өлчөмдүү массивдин элементтерине кокус образ менен маани берүү. Бул үчүн кокус сандарды эсептөөчү санакчыны кошушубуз зарыл.

```
for i:=1 to 20 do b[i]:=random(n);
{n дин мааниси алдын-ала берилиши керек}
```

- бир өлчөмдүү массивдин элементтерине формула боюнча маани берүү.

```
for i:=1 to 20 do b[i]:=5*i*i+4*i-7;
```

2. Бир өлчөмдүү массивдин маанилерин экранга чыгарууда да цикл уюштурабыз.

```
for i:=1 to 20 do writeln(b[i]);
```

3. Бир өлчөмдүү массивдин элементтеринин суммасын эсептөө

```
s:=0;
for i:=1 to 20 do s:= s+b[i];
```

4. Берилген шартты канааттандырган бир өлчөмдүү массивдин элементтеринин санын эсептөө.

Мейли, бир өлчөмдүү массивдин оң элементтеринин санын эсептөө талап кылынсын.

```
k:=0;
for i:=1 to 20 do if b[i]>0 then k:=k+1;
writeln('оң элементтердин саны=',k);
```

5. Бир өлчөмдүү массивдин максималдык(минималдык) элементин жана анын катар номерин аныктоо

Бир өлчөмдүү массивдин максималдык(минималдык) элементин табуу үчүн массивдин элементтерин бири-бир менен салыштырабыз. Максималдык(минималдык) элементти издөө алгоритмин эки түгөй сандарды салыштыруу жана бул салыштыруу аракетин керектүү санда кайталоо аркылуу түзүп алсак болот.

Демек, биз эки суроого жооп издешибиз керек:

- салыштыруу опреациясын түзгөн түгөйлөргө кандай сандар кирет?
- салыштыруу операциясын канча жолу кайталоо зарыл?

Ысмы max болгон кошумча өзгөрмөнү киргизели. Ал биз издеп жаткан түгөй сандардын бири болуп эсептелет, ал эми экинчиси массивдин кийинки кезектеги элементи. Биринчи салыштыруу аракетин жүргүзүү үчүн max өзгөрмөсүнө массивдин биринчи элементин ыйгарабыз, анда салыштыруу аракетинин кайталоо саны n-1 болот.

1-Мисал. Бир өлчөмдүү массивдин максималдык элементин табуу.

```
Program max;
type mas=array[1..20] of integer;
var b:mas;i:byte;max:real;
begin {маани берүү блогу}
for i:=1 to 20 do readln(b[i]);
{максималдуу элементти издөө}
max:=b[1];
for i:=2 to 20 do if b[i]>max then max:=b[i];
writeln('максималдык элемент=',max);
end.
```

2-Мисал. Бир өлчөмдүү массивдин жуп элементтеринин минималдуу элементин тапкыла.

Түшүндүрмө: биз min өзгөрмөсүнө массивдин биринчи элементин ыйгара албайбыз. Себеби ал элемент так болуп калышы мүмкүн. Ошондуктан берилген тип боюнча кандайдыр бир чоң санды тандап алышыбыз зарыл. Эгер биз массивдин элементтеринин тибин integer деп баяндасак, анда мындай сан болуп +32767 саны эсептелет.

```
Program min;
type mas=array[1..20] of integer;
var b:mas; i:integer; min:integer;
```

```
begin {маани берүү блогу}
for i:=1 to 20 do readln(b[i]); min:=32767;
 for i:=1 to 20 do
  if (b[i]<min) and (b[i] mod 2=0) then min:=b[i];
   if min=32767 then writeln('массивде жуп элемент-
   тер жок') else writeln('жуп элементтердин минимал-
   дык элементи=',min);
end.
```

ЭКИ ӨЛЧӨМДҮҮ МАССИВДЕР

Математикада кээде көп өлчөмдүү массивдерди, б.а. массивдердин массивин колдонушат. Абдан кеңири таралгандары бөлүп эки өлчөмдүү массивдер эсептелишет. Мындай массивдерди таблица түрүндө көрсөтсөк болот. Таблицанын ар бир элементи эки индекске ээ, алар элементтин жайгашуу ордун (координатасын) аныктайт.

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix}$$

Математикада квадраттык жана тик бурчтуу таблицаларды кээде матрицалар деп аташат. Биринчи индекс – бул сапчанын номери, ал кийинки сапка өткөндө гана өзгөрөт. Экинчи индекс – мамычанын номери. Информатикада ар бир таблица эки өлчөмдүү массив катары каралат. m сапчадан жана n мамычадан турган матрицаны mxn өлчөмдө деп айтышат. Эки өлчөмдүү массивдерди колдонуп маселелерди чыгарууда кийиштирилген циклдар уюштурулат.

Сапча боюнча жылдыруу:

```
for i:=1 to m do сырткы цикл, сапчанын номери өзгөрөт
for j:=1 to n do ички цикл, мамычанын номери өзгөрөт
...
```

Мамыча боюнча жылдыруу:

```
for j:=1 to n do   сырткы цикл, мамычанын номери өзгөрөт
for i:=1 to m do   ички цикл, сапчанын номери өзгөрөт
...
```

Эки өлчөмдүү массивдерди иштетүүгө арналган базалык алгоритмдерди санап өтөлү:

1. Эки өлчөмдүү массивдин элементтерине маани берүү
2. Таблица түрүндө чыгаруу
3. Ар бир сапчанын жана мамычанын элементтеринин суммасын эсептөө
4. Массивдин элементтеринин суммасын эсептөө
5. Ар бир сапчанын(мамычанын) максималдуу(минималдуу) элементтин жана алардын индекстерин табуу
6. Массивдин максималдуу(минималдуу) элементин табуу

Базалык алгоритмдерди кароодон мурун, эки өлчөмдүү массивдин тибин баяндоону карап өтөлү. Массивди баяндоо төмөнкү формада көрсөтүлөт:

```
type mas=array[1..3,1..4] of integer;
var a:mas;
```

Бул жерде mas – типтин аты, ал эми a – үч сапчага жана төрт мамычага ээ эки өлчөмдүү массив. Эки өлчөмдүү массивди башкача жол менен да баяндаса болот. Мейли m=array[k_1..k_2] of <базалык тип> болсун, мында базалык типтин катарында real, integer, char, ж.б. стандарттык типтеринен сырткары, мурда көрсөтүлгөн тип б.а. массивдердин массиви алынат.

```
type m3=array[1..4] of integer;
     m4=array[1..3] of m3;
var a:m4; {эки өлчөмдүү массив}
    b:m3; { бир өлчөмдүү массив}
```

- Эки өлчөмдүү массивдин элементтерине маани берүү.

  Массив 3 сапчадан жана 4 мамычадан б.а. 3x4=12 элементтен турат. 1-сапчанын элементтерине маанилерди берели:

```
for i:=1 to 3 do
for j:=1 to 4 do
  a[i,j]:=random(100);
```

  Экинчи сапчанын элементттерине маани берүү үчүн сапчанын индексин 1 ге өзгөртүп, 2 ни алабыз ж.б. ушундай жол менен калган элементтердин маанилерин беребиз. Сапчанын индекси мамычанын индексине караганда жай өзгөрөт.

  Ошондой эле read операторунун жардамы менен клавиатурадан маани беребиз:

```
for i:=1 to 3 do
for j:=1 to 4 do read(a[i,j]);
```

- Берилген эки өлчөмдүү массивдин элементтерин таблица формасында чыгаруу.

```
for i:=1 to 3 do begin
for j:=1 to 4 do {курсорду жылдырбай туруп
сапчанын элементтерин чыгарабыз}
 write(a[i,j],' '); writeln; {курсорду которуу}
 end.
```

- Ар бир сапчанын жана мамычанын элементтеринин суммасын эсептоо

```
for i:=1 to 3 do
begin s:=0;
      for j:=1 to 4 do s:=s+a[i,j]; writeln(s,i);
end;
```

■ Массивдин элементтеринин суммасын эсептөө

```
s:=0;
for i:=1 to 4 do
for j:=1 to 4 do s:=s+a[i,j];
```

■ Ар бир сапчанын(мамычанын) максималдуу(минималдуу) элементин жана алардын индекстерин табуу

Ар бир сапчанын максималдуу элементин табуу маселесин бир өлчөмдүү массивдин максималдуу элементин табуу базалык алгоритмин бир нече жолу кайталоо менен чыгарсак болот.

а) i-сапча үчүн алгоритм төмөнкүдөй түрдө болот:

```
max:=a[<саптын номери>,1];
for j:=1 to n do if a[<саптын номери>,j]>max then
    max:= a[<саптын номери>,j];
```

Эми алгоритмди m жолу кайталайлы:

```
for i:=1 to m do
begin max:=a[i,1];
 for j:=1 to n do if a[i,j]>max then max:=a[i,j];
end;
```

Алгоритмди максималдык элементтин индексин көрсөтүү менен иштетели.

```
for i:=1 to m do
begin max:=a[i,1];
ind_L:=i;    {саптын номерин сактоо}
ind_C:=1;    {1-мамычанын номерин киргизүү }
for j:=1 to n do
if a[i,j]>max then begin max:= a[i,j];
ind_C:=j;    {j-мамычанын номерин сактоо}
end; writeln(i,'-саптын max=',max);
end;
```

б) минималдуу элементти табуу жана ар бир мамыча үчүн анын индекстерин көрсөтүү алгоритмин карап көрөлү.

```
for j:=1 to n do      {мамыча боюнча}
begin min:=a[1,j];
ind_L:=1;             {саптын номерин сактоо}
ind_C:=j;          {мамычанын номерин киргизүү }
for i:=1 to m do
if a[i,j]>min then begin min:= a[i,j];
ind_L:=i;          {i-саптын номерин сактоо}
end; writeln(j,'-мамычанын min=',min);
end;
```

■ Массивдин максималдуу(минималдуу) элементин жана алардын индексин табуу

```
min:=a[1,1]; ind_L:=1; ind_C:=1;
for i:=1 to n do
```

```
for j:=1 to m do
if a[i,j]<min then begin
min:=a[i,j]; ind_L:=i; ind_C:=j;
end;
```

КВАДРАТТЫК МАТРИЦАЛАР

Сапчаларынын саны менен мамычаларынын саны дал келген эки өлчөмдүү массив квадраттык матрица деп аталат. Аны баяндоо төмөндөгүдөй болот:

```
type mas=array[1..4,1..4] of integer;
var a:mas;
```

a_{11}	a_{12}	a_{13}	a_{14}
a_{21}	a_{22}	a_{23}	a_{24}
a_{31}	a_{32}	a_{33}	a_{34}
a_{41}	a_{42}	a_{43}	a_{44}

Квадраттык матрицанын башкы диогоналдык элементтери a_{11}, a_{22}, a_{33}, a_{44} (i=j), болуп эсептелет. Сүрөттө башкы диогоналдын элементтери боз түс менен берилген. Тескери диогоналдарынын элементи болуп a_{41}, a_{32}, a_{23}, a_{14} эсептелет б.а. $i + j = 4 + 1$ же жалпы түрдө $i + j = n + 1$ шарты аткарылышы зарыл. Тескери диогоналынын элементтери сүрөттө кара түстө көрсөтүлгөн. Башкы диогоналдын үстүндө жайгашкан элементтердин индекси үчүн $i < j$ катышы орун алат. Башкы диогоналынын астында жайгашкан элементтердин индекстери үчүн i>j аткарылышы керек.

1-Мисал. Башкы диогоналдык элементтердин суммасын эсептегиле.

```
S:=0;for i:=1 to n do S:=S+a[i,j];
```

2-Мисал. Тескери диогоналынын минималдык элементин тапкыла.

```
min:=a[1,n];
for i:=1 to n do
if a[i,n+1-i]<min then min:=a[i,n+1-i];
```

3-Мисал. Матрицаны транспонирлегиле

```
for i:=1 to n do
for j:=1 to n do
b[i,j]:=a[j,i];
```

3. ПРОЦЕДУРАЛАР ЖАНА ФУНКЦИЯЛАР

САПТЫК ЧОҢДУКТАР

Лексикалык жана синтаксистик маселелерди чыгарууда, биз литердик типтеги маалыматтарды колдонобуз. Литердик тип Паскаль тилинде STRING сапчасы менен берилет.

STRING тибинин өзгөрмөсү 0 дөн N ге чейинки символдорду камтыйт, бирок 255 тен ашпайт. Саптын узундугун STRING кызматчы сөзү менен катар квадраттык кашаада көрсөтүүгө болот.

```
var  st:string[30];
```

Эгерде саптын узундугу көрсөтүлбөсө, анда ал максималдуу б.а. N=255 маанеге ээ болуп калат. Саптар символдордон түзүлгөн чынжырча катарында каралат.

String тиби көп жагынан бир өлчөмдүү масивдерге окшош. Бирок символдор массивинен айырмаланып, саптагы символдордун саны 0 дөн N ге чейин өзгөрөт. Саптагы каалаган символго массивдин элементине кайрылган сыяктуу кайрылсак болот, б.а. string тибиндеги өзгөрмөнүн аты менен катар квадраттык кашаада саптагы символдун индексин көрсөтөбүз.

st[2], б.а. st сабындагы 2-символ

st[i] - st сабындагы i-символ

Саптар менен төмөндөгүдөй операцияларды жүргүзөбүз:

1. Саптарды бириктирүү операциясы(конкетанация). Бул операция "+" (бул математикадагы кошуу белгиси эмес) белгиси менен белгиленет.

```
st:='компью'; st1:='тер'; st2:=st+st1;
```

жыйынтыгы: st2='компьютер'

2. Салыштыруу операциялары: =; >=; >; <>; <; <=. Ар кандай узундуктагы саптарды салыштырууга болот. Салыштыруу ASCII кодуна ылайык солдон оңду көздөй жүргүзүлөт.

string маалыматтар тиби тексттерди иштетүүдө колдонулат. Бул болсо бизге:

- саптын белгиленген бөлүгүн көчүрүүнү;
- саптын белгиленген бөлүгүн өчүрүүнү;
- саптын белгиленген бөлүгүн көрсөтүлгөн санка коюуну;
- саптын белгиленген бөлүгүн көрсөтүлгөн саптан издөөнү

аткарууга мүмкүндүк берет.

Бул операцияларды аткаруу үчүн Паскаль тилинде стандарттык процедуралар жана функциялар орун алган.

1. Саптын белгиленген бөлүгүн көчүрүп алуу – copy(st,n,k) функциясы, мында st – көрсөтүлгөн сап, n – баштапкы позиция, k – ко

чүрүү үчүн зарыл болгон символдордун саны

```
st:='мектеп'; st1:=copy(st,4,3);
```

st1 сабынын мааниси - 'теп'

2. Саптын белгиленген бөлүгүн өчүрүү - delete(st,n,k) процедурасы, мында st - көрсөтүлгөн сап, n - баштапкы позиция, k - өчүрүлүүчү символдордун саны

```
st:='китеп'; delete(st,2,2);
```

st сабынын мааниси - 'кеп'

Көңүл бургула! Саптын белгиленген бөлүгүн өчүрүүдө узундугу авто-маттуу түрдө кичирейет: st[0]=3.

3. Көрсөтүлгөн сапка белгиленген саптын бөлүгүн коюу - insert(st1,st,n) процедурасы, мында st1 - белгиленген саптын бөлүгү, st - көрсөтүлгөн сап, n - ордуна кою башталуучу символдордун позициясы.

```
st:='арча'; st1:='ал'; insert(st1,st,3);
```

st сабынын мааниси - 'аралча'

4. Белгиленген саптын бөлүгүн көрсөтүлгөн саптан издөө - k:=pos(st1,st) функциясы, мында st1 - белгиленген саптын бөлүгү, st - көрсөтүлгөн сап, k - белгиленген саптын бөлүгүнө алгачкы кирүү позициясы

1	2	3	4	5	6	7	8	9	10	11	12	13	14
А	С	А	Н		У	У	Л	У		Э	С	Е	Н

'УУЛУ' белгиленген сабын st:='АСАН УУЛУ ЭСЕН' көрсөтүлгөн сабынан издейбиз: k:=pos('УУЛУ',st);

Жообу: k=6.

Көңүл бургула! Дагы бир жолу кайталайлы:

- pos функциясынын мааниси - бүтүн типтеги өзгөрмө;
- pos функциясы белгиленген саптын биринчи жолуккан позициясын көрсөтүлгөн саптан издейт

string тибиндеги сап тамга жана башка символдор сыяктуу эле сандарды да өз ичине камтыйт.

```
var st,st1,st2:string;
begin
     st:='123';st1:='254';
     st2:=st+st1;
     writeln(st2);
end.
```

st2 сабынын мааниси 123254

```
var a,b, k:integer;
begin
     a:=123;b:=254;
     k:=a+b;
     writeln(k);
end.
```

k=377

Жогоруда каралган мисалдарды салыштырып көрөлү. 1-мисалда символдордун(сандардын) тиби string жана бул жерде саптарды бириктирүү операциясы жүргүзүлдү. Ал эми 2-мисалда болсо өзгөрмөнүн

тиби integer жана бул жерде математикадагы кошуу амалы жүргүзүлдү.

5. length функциясы саптын узундугун көрсөтөт.

```
...
st:='Кыргызстан'; k:=length(st); writeln(k);
...
```

Жообу: k=10.

Сан маалыматын литердик чоңдуктарга айлантуу жана литердик чоңдуктарды сан маалыматына айландырууга багыттал-ган көптөгөн маселелер менен иштөөгө туура келет. Бул проблемаларды чечүү үчүн Паскаль тилинде төмөнкүдөй процедуралар каралган:

1. STR процедурасы каалаган анык же бүтүн типтеги санды символдор сабына айландырат: str(b,st);

Мында: b - сан өзгөрмөсү, st - символдор сабы. Мисалы:

```
var b:integer; c:real; st,st1:string;
...
b:=1978; str(b,st);
c:=14.7531; str(c,st1);
...
```

st сабынын мааниси - '1978'

st1 сабынын мааниси - '1.4753100000E+01'

2. VAL процедурасы цифраларды камтыган сапты анык же бүтүн типтеги санга айландырат: val(st,b,code);

Мында: b - сан өзгөрмөсү, st - символдор сабы, code - бүтүн типтеги өзгөрмө, анын мааниси боюнча алмаштыруу кандай жүргөнүн билүүгө болот. Эгер символдор сабынан санга алмаштыруу процесси ийгиликтүү өтсө, анда code өзгөрмөсүнүн мааниси 0гө барабар болот.

```
var b,code:integer; c:real; st,st1:string;
...
st:='1978'; val(st,b,code);
st1:='14.7531'; str(st1,c,code);
...
```

b өзгөрмөсүнүн мааниси b=1978

c өзгөрмөсүнүн мааниси c=1.4753100000E+01

1-мисал. Сап жалгыз гана сандардан турат. Алдыда турган бардык 1 санын өчүргөлө.

Көрсөтүлгөн сап: 11111164128351608

Жыйынтык: 64128351608

```
program del;
var st:string; i: byte;
begin
writeln('сапты киргизгиле'); readln(st);i:=length(st);
while ((i>1) and (st[1]='1')) do  begin
```

```
  delete(st,1,1); i:=i-1;
  end;
writeln(st);
end.
```

2-мисал. st символдор сабы сандардан гана турат. Бул сан кандайдыр p-эсептөө системасында жайгашкан. Бул санды ондук эсептөө системасына которгула.

p-эсептөө системасынан ондук эсептөө системасына которууну карайлы. Мейли 1978 саны 5-эсептөө системасындагы сан болсун. Аны ондук эсептөө системасына которобуз:

$$1978_5 = 1\cdot5^3 + 9\cdot5^2 + 7\cdot5^1 + 8\cdot5^0 = 8\cdot5^0 + 7\cdot5^1 + 9\cdot5^2 + 1\cdot5^3$$

Бул жерден биз эки базалык алгоритмди колдонсок жетиштүү болот:

- сумманы эсептөө
- a^n ди эсептөө

Мейли s өзгөрмөсү сумманы топтоо үчүн, p өзгөрмөсү 5^n ди эсептөө үчүн колдонулсун. Анда программанын фрагменти төмөнкү түрдө болот:

```
...
s:=0; p:=1;
for i:=lenght(st) downto 1 do  begin
{цифрадан турган символду санга которуу}
  val(st[i],a,code);
  p:=p*5; s:=s+p*a;
  end;
...
```

КАМТЫЛГАН ПРОГРАММАЛАР

Камтылган программа – кандайдыр бир камтылган маселени чыгарууга арналган операторлордун удаалаштыгы. Ар бир камтылган программанын ысмы бар жана ал боюнча ага кайрылууга болот.

Төмөнкү маселени карайлы.

15 элементтен турган бир өлчөмдүү массивди бүтүн сандар менен кокус образда толтургула жана

1) Массивдин бардык элементтеринин;
2) 3-дөн 7-ге чейинки элементтеринин;
3) 7-дөн 13-го чейинки элементтеринин суммаларын эсептегиле.

Саналып өткөн циклдерди деталдаштыралы:

- массивге маани берүү блогу

```
for i:=1 to 15 do
a[i]:=random(100);
```

- массивдин бардык элементтеринин суммасын эсептөө

```
S:=0;
for i:=1 to 15 do S:=S+a[i];
```

■ массивдин 3-дөн 7-ге чейинки элементтеринин суммасын эсептөө

```
S:=0;
for i:=3 to 7 do S:=S+a[i];
```

■ массивдин 7-ден 13-гө чейинки элементтеринин суммасын эсептөө

```
S:=0;
for i:=7 to 13 do S:=S+a[i];
```

Жогоруда көрүнүп тургандай биз сумманы эсептөө алгоритмин 3 жолу кайталадык, бул жерде i өзгөрмөсүнүн баштапкы жана акыркы маанилери гана өзгөрдү.

Программалоодо бир эле аракетти бир нече жолу кайталоого туура келген учурлар кездешет. Биз негизги маселелерден бөлүнүп көрсөтүлгөн камтылган маселени чыгаруучу кандайдыр алгоритмге ээ болобуз. Камтылган маселени чыгаруу алгоритмин программанын керектүү жерине улам-улам кайталап жаза берүү рационалдуу эмес болуп калат. Паскаль программалоо тили камтылган маселени чечүүчү каалаган алгоритмди бөлүп алууга, ага ысым берип, аны бир жолу жазууга жана канча жолу керек болсо, ошончо жолу колдонууга мүмкүндүк берет.

Паскалда камтылган программалар процедураларга жана функцияларга бөлүнөт. Алар программанын объектиси болуп эсептелишет жана маалыматтарды баяндоо бөлүгүндө жайгашат:

```
PROGRAM main; {негизги программанын аталыш аймагы}
    TYPE mas=array[1..15] of integer;
    VAR  a:=mas; i:byte;
{процедуранын же функциянын аталыш аймагы, өзгөрмөлөрдү
баяндоо бөлүгү}
    BEGIN {камтылган программанын башы}
    {камтылган программанын түзүлүшү}
    END; {камтылган программанын аягы}
BEGIN {негизги программанын башы}
    {негизги программанын түзүлүшү}
END. {негизги программанын аягы}
```

Өз кезегинде камтылган прорамманын түзүлүшүндө да камтылган программалар жайгаша алат.

ГЛОБАЛДЫК ЖАНА ЛОКАЛДЫК ӨЗГӨРМӨЛӨР

Сиздер билесиздер, негизги программанын түзүлүшүндө өзгөрмөлөрдү баяндоо бөлүгү жайгашкан. Камтылган программанын түзүлүшү да, эгер алар зарыл болуп жатса, өзгөрмөлөрдү баяндоо бөлүгүн ичине алат. Бирок өзгөрмөлөр кездешпеген маселелер чанда гана кездешет, ошондуктан аларга түзүлгөн программада өзгөрмөлөрдү баяндоо бөлүгүн эске албай койсок да болот.

```
    PROGRAM main; {негизги программа}
    TYPE mas=array[1..15] of char;
    VAR  a:=mas;i,j:byte;
{камтылган программанын аталыш аймагы}
{маалыматтарды баяндоо бөлүгү,төмөнкү кызматчы сөздөрдү
камтышы мүмкүн: type,const,var,procedure,function}
    TYPE m:array[1..10] of real;
    VAR b:m; k:byte;
    BEGIN {камтылган программанын башы}
    {камтылган программанын операторлору}
    END;
BEGIN {негизги программанын башы}
    {негизги программанын түзүлүшү}
END. {негизги программанын аягы}
```

Программанын аткарылышында анын негизги бөлүгүндө баяндалган өзгөрмөлөр **глобалдык өзгөрмөлөр** деп аталат. Аларды каалаган камтылган программада колдонууга болот. Жогорку мисалда көрсөтүлгөн глобалдуу өзгөрмөлөр:

```
a (тиби - mas), i,j ( тиби - byte)
```

Камтылган прорамманын аткарылышында баяндалган өзгөрмөлөр **локалдык** деп аталат. Анткени бул өзгөрмөлөр камтылган программанын аткарылышы үчүн гана кызмат кылат. Негизги программа аларды "байкабайт". Жогорку мисалда көрсөтүлгөн глобалдуу өзгөрмөлөр: `b (тиби m- массив), k (тиби - byte)`

Көңүл бургула! Глобалдык өзгөрмөлөр каалаган камтылган программа үчүн тийиштүү, ал эми локалдык өзгөрмөлөр кайсы камтылган программада баяндалса, ошого гана тийиштүү.

ПРОЦЕДУРАЛАР

Процедуралар атайын эрежелер боюнча жазылууга тийиш. Процедуранын аталыш аймагы `procedure` кызматчы сөзүн жана процедуранын ысмын камтыйт, алар Паскаль тилинин эрежелерине баш ийишет. Жазылыш форматы төмөнкүдөй:

```
procedure proc(<кызматчы маалымат - параметрлер>)
```

Өзгөрмөлөрдү баяндоо бөлүгү жана кызматчы маалымат колдонулбашы мүмкүн, ал эми операторлор милдеттүү түрдө `begin` кызматчы сөзү менен башталып, `end` кызматчы сөзү менен аякташы керек. Демек милдеттүү болуп төмөнкү саптар эсеп-телинет:

```
procedure p;
begin
...
end;
```

Жалпы учурда операторлук кашаанын ортосунда камтылган программанын түзүлүшү жайгашат. Процедуралар негизги программадан өздөрүнүн ысымдары боюнча чакырылат.

Эми биз процедуранын ысмынын жанындагы тегерек кашаанын ичинде кандай кызматчы маалымат жайгашканын аныктайлы.

Камтылган программа негизги программа тарабынан чакырылган кандайдыр бир жекече алгоритмди иштетет. Камтылган программадагы алгоритмди иштетүү үчүн негизги программадагы кээ бир маалыматтар керек болуп калат. Эгерде алар зарыл болсо, анда бул маалыматтарды иштетип, жыйынтыгын негизги программага кайтарат. Негизги программадан камтылган программага информацияны берүү жана тескерисинче камтылган программадан камтылган негизги программага информацияны берүү **параметрлер** деп аталган өзгөрмөлөр аркылуу гана жүргүзүлөт. Схемалык жактан карасак сол жакта көрсөтүлгөн сүрөттөгүдөй болот.

Негизги программадан камтылган программага берүүдөгү өзгөрмөлөр **кирүүчү параметрлер** деп, ал эми информацияны камтылган программадан негизги программага берүүдөгү өзгөрмөлөр **чыгуучу параметрлер** деп аталышат. Параметрлер локалдык болуп эсептелишет жана камтылган программанын иштөө мезгилинде гана мааниге ээ болушат. Бул өзгөрмөлөр процедуранын ысмынын жанында тегерек кашаада көрсөтүлөт. Мунун өзү аларды баяндоо үчүн жетиштүү болот.

Көңүл бургула! Бул өзгөрмөлөр мындан ары программанын же камтылган программанын кандай гана бөлүгүндө болбосун баяндалбайт.

Камтылган программага келип түшкөн информацияларды мүөзөөчү өзгөрмөлөр **параметр-маанилер** деп аталышат.

```
Procedure proc(a,b,c:integer);
```

Процедурага болгон ар бир кайрылуунун алдында параметрлерге кандайдыр бир маанилерди беришибиз керек. Параметр тизмеси үтүрлөр аркылуу бөлүнүп берилет жана алар өзгөрмөлөрдүн санын формалдуу түрдө аныкташат.

1-Мисал. b^k ны эсептөөчү процедураны жазгыла, мында k – натуралдык сан. Негизин - b, даража көрсөткүчүн k деп алалы.

```
{негизги программанын өзгөрмөлөрүн баяндоо бөлүгү}
VAR b:real; {негизи - глобалдык чоңдук}
    k:integer; {даража көрсөткүч}
    P:real; {жыйынтык}
Procedure pw(a:real;n:integer); {a жана n - локалдык
                                 өзгөрмөлөр}
var i:byte; {циклдын эсептегичи - локалдык өзгөрмө}
begin
    P:=1; for i:=1 to n do P:=P*a;
end;
BEGIN {негизги программанын башы}
readln(b,k); {b - негизи, k- даража көрсөткүчү}
pw(b,k); writeln(b,'нын',k,'-даражасы=',P);
END.
```

Негизги проограмманын жана камтылган программанын түзүлүшүндө колдонулган бардык өзгөрмөлөрдү топтоштуралы.

Глобалдык өзгөрмөлөр: b, k, P – бардыгы үчүн кызмат кылат. Локалдык өзгөрмөлөр: a, n, i – процедура үчүн гана кызмат кылат.

Параметрлердин берилишине кеңенирээк токтололу:

- Процедуранын ысмынан кийинки кашаадагы параметрлер формалдуу болуп эсептелишет. Биздин максат – информацияны процедурага берилишиндеги жардам кылган өзгөрмөлөрдүн санын жана тибин көрсөтүү.
- Процедурага кайрылганда параметрлердин мааниси даана такталышы керек, кандайдыр маанини даана так көрсөтүү керек (мында маанилер тизмеси же формалдуу параметрлер тизмеси процедуранын аталыш аймагында кандай берилсе, ошондой тартипте берилет) же аларга кандайдыр бир маанини ыйгарып, өзгөрмолөрдүн ысмын санап өтүү керек.
- Процедураны чакырууда тегерек кашаадагы маанилердин же өзгөрмөлөрдүн тизмеси фактылык параметрлердин тизмеси болуп саналат.

2-Мисал.

```
program p;
var a,c:integer;{глобалдык өзгөрмөлөр}
   procedure proc(b,d:integer);
     var i:byte;
     begin
     ... {процедуранын түзүлүшү}
     end;
begin
a:=3; c:=4; proc(a*c,a*c-10); {процедураны чакыруу}
end.
```

Көңүл бургула! Эгер параметрлер параметр-маанилер болуп эсептелишсе, анда кандайдыр бир туюнтма менен берилген маа-нини берүүгө болот. Бирок бул жерде өзгөрмөлөр белгилүү болушу керек. Камтылган программага эсептелинген туюнтманын мааниси берилет.

b=3*4=12, d=3*4-12=2

Өзгөрмөлөрдүн мүнөздөмөлөрү: a, c - глобалдык өзгөрмөлөр
фактылык параметрлер: a*c, a*c-10; i, b, d - локалдык өзгөрмөлөр
b, d - формалдык параметрлер

Камтылган программа иштетүүчү информация 2 топко бөлүнөт: киргизилүүчү информация жана чыгарылуучу информация.

Киргизилүүчү информацияны иштетүүдө анын маанилери камтылган программага параметр-маанилер аркылуу келип түшөт. Негизги программа жыйынтыкты, б.а. чыгарылуучу информация-ны алат. Бул учурда биз парметр-өзгөрмөлөрдү колдонсок болот. Алар параметрлер тизмесинде жооптун мааниси ыйгарылуучу өзгөрмөнү көрсөтөт.

Процедураны чакыруу учурунда параметр-өзгөрмолөр компьютердин эсинен оруп бөлүнгөнүнө карабай мааниге ээ болушпайт. Алар мааниге ээ болушу үчүн, менчиктөө операторунун жардамы менен жыйынтыктын маанисин аларга ыйгаруу керек.

Эсептөө жыйынтыгы негизги программага парметр-өзгөрмөлөр аркылуу келип түшкөндөй кылып a^n ди эсептеген pw процедурасын өзгөртөлү.

```
Program p;
var a:real; {негизи}
    n:byte; {даража көрсөткүч}
    x,y:real; {жыйынтык}
    procedure pw(b:real;k:byte;var c:real);
        var i:byte;
        begin
            c:=1; for i:=1 to k do c:=c*b;
        end;
begin {негизги программа}
```

```
pw(7.2,5,x); writeln(x:3:4);
pw(-2.3,4,y); writeln(y:3:4);
{x,y - фактылык өзгөрмөлөр, алардын жардамы менен
жыйынтык негизги программага белгилүү болот}
end.
```

ФУНКЦИЯЛАР

Функция - маанини кайтаруучу камтылган программа. Баяндалуу форматы төмөнкүдөй:

```
function<ысмы>(формалдуу параметрлер тизмеси)
  {локалдуу өзгөрмөлөрдү баяндоо бөлүгү}
begin
    ... {функциянын түзүлүшү}
```

Функция үчүн милдеттүү түрдө тиби көрсөтүлөт. Мисалы: n! ды эсептөө.

```
Var n:byte;
{глобалдуу өзгөрмө}
ff:longint;
function fac(k:byte):longint;
{longint-жыйынтыктын{функциянын маанисинин} тиби}
  var i:byte; F:longint;
begin
F:=1; for i:=1 to k do F:=F*i;
fac:=F;{милдеттүү оператор, функцияга маани менчиктелет}
end; {функциянын аягы}
begin {негизги программанын башы}
writeln('n ди киргиз'); readln(n);
ff:=fac(n); writeln('n!=',ff);
end.
```

Өзгөрмөлөрдүн мүнөздөмөлөрү:

n, ff – глобалдык өзгөрмөлөр; фактылык параметрлер: n

k, i, f – локалдык өзгөрмөлөр; k – формалдык параметрлер

Камтылган программалар боюнча алган билимдерибизди жыйынтыктайлы. Демек, эгер алгоритм өзүнүн багытталышы боюнча бир нече жолу кайталануучу бөлүктөрдү өз ичине камтыса, анда камтылган программалар колдонулат. Паскаль тилинде камтылган программанын 2 түрү менен тааныштык:

Процедуралар	Функциялар
procedure	function

Камтылган программанын түзүлүшү өзгөрмөлөрдү баяндоо бөлүгүндө жайгашкан. Негизги программада баяндалган өзгөрмөлөр

глобалдык деп аталышат да, каалаган камтылган программада колдонулуу мүмкүнчүлүгүнө ээ болот. Камтылган программада баяндалган өзгөрмөлөрдү камтылган программанын өзүнө гана колдоно алабыз, камтылган программанын иши аяктагандан кийин бул өзгөрмөлөрдү колдонууга мүмкүнчүлүк берилбейт.

Информацияны негизги программадан камтылган программага берүү үчүн параметрлер деп аталган өзгөрмөлөр колдонулат. Параметрлер формалдуу – процедуранын аталыш аймагындагы кашаанын ичиндеги параметрлер жана фактылык – камтылган программаны чакыруудагы параметрлердин тизмеси болуп бөлүнөт.

Ошондой эле параметрлер: параметр-маанилер жана параметр-өзгөрмөлөр болуп бөлүнөт. Алардын аткарган милдети менен жогоруда тааныштык.

1-Мисал. x жана y аныктары берилген.

$f(x,y)$ тиэсептегиле. Мында $f(a,b)=\frac{a+b^2-\cos ab}{1.8+|a-b|}$

```
program f1;
var x,y:real;
function f(a,b:real):real;
begin f:=(a+sqr(b)-cos(a*b))/(1.8+abc(a-b)); end;
begin {негизги программа}
  writeln('x,y ти киргиз'); readln(x,y);
  writeln('f функциясынын маанисиси=',f(x,y));
end.
```

2-Мисал. a жана b сандарынын массивдери берилген, ар бир массивдин элементтеринин саны 12. Массивдерди кокус образ менен толтургула. U=max(a_1, a_2, a_3, ..., a_{12})+max(b_1, b_2, b_3, ... b_{12}) ни эсептегиле.

```
program f2;
type mas=array[1..12] of real;
var a,b:mas; i:byte;
  function max_mas(c:mas):real;
  var  j:byte; m:real;
  begin
   m:=c[1];
   for j:=1 to 12 do if m<c[j] then m:=c[i];
   max_mas:=m;
  end; {функциянын аягы}
BEGIN {негизги программа}
for i:=1 to 12 do begin
 a[i]:=random/1.2-34.7; b[i]:=57.5-random/3.4; end;
write('максималдык элементтердин суммасы');
writeln(max_mas(a)+max_mas(b));
END.
```

4. ФАЙЛДАР МЕНЕН ИШТӨӨ

ПАСКАЛЬ ТИЛИНИН ФАЙЛДАРЫНА БОЛГОН БАГЫТ

Файл - маалыматтардын көптүгүнүн аталышы. Бул болсо программага бир нече файлдар менен иштөөгө мүмкүндүк берет, б.а. ар бир файл өзүнчө ысымга ээ. Ошондой эле маалыматтардын көптүгүн башкалардан айырмалап турат. Файлдын компоненттери бир типтүү болушу керек. Паскаль тилинде компоненттердин файл тибинен башка каалаган тибин колдонууга болот, б.а. файлдардан турган файл түзүүгө болбойт. Түзүлүп жаткан файлдын узундугу(компоненттердин саны) файлды түзүүдө баяндалбайт. Ал сырткы эске тутуучунун физикалык мүмкүнчүлүгү менен чектелет.

Файлдын ысмы - бул 8 символду камтыган (пробел, чекит, үтүр, кош чекит, жылдызча, суроо, тескери жантык сызык символдорунан башка) саптык өзгөрмө.

Файлдын ысмы мүмкүндүк берилген каалаган символдон башталат. Файлдын атынан кийин анын кеңейтилиши чекит аркылуу ажыратылып берилет. Мисалы: L1.txt, мында L1 – файлдын негизги ысмы, txt – ысымдын кеңейтилиши.

Шартка ылайык ысымдын астында файлга болгон жол – дисктин аты, учурдагы каталогдун аты, камтылган каталогдун аты көрсөтүлөт. Мисалы: `c:\TURBO\BIN\turbo.exe`

Файлды түзүүдө төмөнкү операциялардын тобун аткаруу керек:

- файлды ачуу;
- файлдын компоненттерин жазуу;
- файлды жабуу;

Файлды окууда төмөнкү операциялардын тобун аткаруу керек:

- окуу үчүн файлды ачуу;
- файлдын компоненттерин окуу;
- файлды жабуу;

Паскаль тилиндеги каалаган программада алдын-ала жарыяланган эки файлды колдонууга болот. Алар менен стандарттык файлдык өзгөрмөлөр: `INPUT` (маалыматтарды клавиатурадан окуу үчүн) жана `OUTPUT`(маалыматтарды экранга чыгаруу үчүн) байланышкан. Паскалдын стандарттык формасында программанын аталыш аймагында бул файлдарды жазуу милдеттүү болуп саналат.

Мисалы: `Program My(input,output);`

Турбо-Паскалда стандарттык файлдык өзгөрмөлөрдү колдонбой койсок деле болот. Ошондой эле Турбо-Паскаль тили программанын аткарылышында түзүлгөн файлдар менен кенири иш алып барат.

Турбо-Паскаль тилинде файлдардын бардык типтерин программада төмөнкү эки операция аткарылгандан кийин гана колдоно берсек болот:

1. Файлдык өзгөрмөнү дисктеги файлдын аты менен байланыштыруу `assign` процедурасы менен жүргүзүлөт.

```
assign(<файлдык өзгөрмөнүн ысмы>,<дисктеги файлдын ысмы>);
```

Мисалы: файлдын тибин жана ушул типке тиешелүү файлдык өзгөрмөнүн тибин баяндайлы.

```
type fl=file of char; var f:fl;
begin
...
assign(f,'c:\L1.txt');
...
end.
```

С дискинде аты L1.txt болгон файл жайланышкан, программада аны менен f файлдык өзгөрмөсү байланышкан.

2. Алмаштыруу багытын көрсөтүү: окуу (файлдан оперативдик эске) же жаздыруу (оперативдик эстен сырткы түзүлүшкө)

Паскаль тилинде файлдардын 3 тибин бөлүп көрсөтүшөт:

- типтешкен файлдар
- тексттик файлдар
- типтешкен эмес файлдар

Файлдардын ар бир тибин баяндоо учурунда кызматчы сөздөр колдонулат.

ТИПТЕШКЕН ФАЙЛДАР

Мындай типтеги файлдар үчүн компоненттин тиби сөзсүз көрсөтүлүшү керек.

```
type fil=file of <тип>;
```

Мында `fil` - типтин аты (колдонуучу көрсөтөт), `file` - тибине карата көрсөтүлгөн кызматчы сөз, `of` - сөздөрдү байланыштыруу үчүн колдонулган кызматчы сөз, `<тип>` - `file` тибинен башка Паскаль тилинин каалаган тибин көрсөтөт.

Мисалы: ысмы `fl` болгон файлдын тибин баяндайлы жана ага `f` файлдык өзгөрмөсүн ыйгаралы

```
type fil=file of integer;
var f:fil;
```

1-сапчаны алып салсак да болот, анда файлдык өзгөрмөнү баяндоо төмөнкүдөй болот:

```
var f:file of integer;
```

Дискте файл түзүү үчүн төмөнкү процедуралар аткарылышы зарыл:

- `assign(f,'<дисктеги файлдын ысмы>');`
- дисктеги файлды ачуу: `rewrite(f);`
- файлдын компонентин жазуу: `write(f,a)`

Мында `f` - файлдык өзгөрмөнүн ысмы, `a` - буфердик өзгөрмө, анын мааниси дискке камтылат. `a` өзгөрмөсүнүн тиби милдеттүү түрдө файлдын компонентинин тиби менен бирдей болушу керек.

- `close(f)` - файлды жабуу

Эгерде мурда дискте жазылган файлды окуу керек болсо, анда аракеттердин удаалаштыгы төмөнкүдөй болот:

- `assign(f,'<дисктеги файлдын ысмы>');`
- окуу үчүн файлды ачуу: `reset(f);`
- файлдын компонентасын окуу: `read(f,a);` Мында `f` - файлдык өзгөрмөнүн ысмы, `a` - буфердик өзгөрмө, ага файл-дык өзгөрмө камтылат.
- `close(f)` - файлды жабуу

Типтешкен файлдар үчүн ар бир компонентага түз аракетти уюштурууга болот, себеби ар кандай компонентанын узундугу туруктуу. Файлдын керектүү болгон компонентасына көрсөткүчтү жылдыруу үчүн `seek` процедурасы колдонулат:

```
seek(<файлдык өзгөрмөнүн ысмы>,n);
```

`n`- компонентанын номери, ал бүтүн типте болот.

Мисалы: бизге файлдын 5-компонентин окуу талап кылынсын.

```
assign(f,'NCG.txt');reset(f);
seek(f,5); read(f,a);
```

Турбо-Паскалда файлдын өлчөмүн аныктоо `filesize` функциясынын жардамы менен жүргүзүлөт.

```
t:=filesize(<файлдык өзгөрмөнүн ысмы>);
```

`t` - бүтүн сандагы өзгөрмө

Корсоткүчтүн жайгашкан ордун аныктоо үчүн Турбо-Паскалда `filepos` функциясы колдонулат:

```
filepos(<файлдык өзгөрмөнүн ысмы>);
```

Жогоруда биз файлдын акыркы компонентасы түзүлгөндөн кийин, файлды жабуу операциясы аткарылаарын айтып өттүк. Турбо-Паскалда файлдын аяктоо белгисин көрсөтүү үчүн `EOF (EndOfFile)` логикалык функциясы ишке жүктөлөт. Ал файл аяктаганда гана чын (туура) деген маанини алат.

1-Мисал. `f` файлынын бардык компоненталарын окуп жана алардын мааниси экранга чыгар.

```
type fil=file of integer; var f:fil; a: integer;
begin
```

```
assign(f,'NCG.txt'); reset(f);
while not eof(f) do begin read(f,a); write(a,''); end;
end.
```

2-Мисал. Компоненттери бир өлчөмдүү массивдер болгон файлды түзүүлө.

```
Program fil_mas;
type mas=array[1..10] of integer; fil=file of mas;
var a:mas; f:fil;i,j:integer;
begin
assign(f,'mas.dat'); rewrite(f);
{файлдын 7 компонентин түзөлү}
for i:=1 to 7 do begin
   for j:=1 to 10 do a[j]:=random(100);
   write(f,a); end; close(f);
end.
```

ТЕКСТТИК ФАЙЛДАР

Файлдык өзгөрмөнү баяндайлы: `var g,f:text;`

Мындай файлдар тексттик информацияларды сактоо үчүн ылайыкташтырылган. Тексттик файлдардын компоненттерин көбүнчө жазылыштар же саптар деп аташат. Саптар өзгөрмө узундукка ээ болушу мүмкүн, бул болсо алар менен иштөөгө таасирин тийгизет. Тексттик файлдарды удаалаш түрдө гана окууга болот.

Тексттик файлдарды түзүүдө ар бир саптын аягында `Enter` баскычын басабыз. Бул команда боюнча система саптын аяктоо белгиси – `eoln` ду коет. Саптын аяктоо белгисин аныктоо үчүн Турбо-Паскалда `eoln (EndOfLine)` функциясы колдонулат.

Сырткы эске тутуучуда файл түзүү үчүн эмне кылуу керек? Бул маселенин чыгарылыш жолдорун издейли.

1-жолу.

Турбо-Паскаль тилиндеги программанын жардамы менен тексттик файл түзөлү. Биздин учурда файл n компонентке ээ деп эсептейли. Компоненттерди жазууну биринчи позицияда '*' белгиси турган сап киргизилгенден кийин токтотолу.

```
Program ncg_text;
var f:text; st:string;
begin
assign(f,'ncg.txt');rewrite(f);
write('саптарды киргиз');
write('ар бир саптын аягында enter ди бас');
while st[1]<>'*' do begin
  readln(st); {сапты клавиатурадан киргизүү}
  writeln(f,st); {сапты файлга жазуу}
```

```
  end; close(f);
end.
```

2-жолу.

Тексттик файлдарды Турбо-Паскалдын тексттик редактору менен түзүү ыңгайлуу. Биз анын кызматы менен программанын текстин жазып жаткандан бери таанышбыз, алардын баары текст-тик файлдар болуп саналышат.

Төмөнкүдөй тартипте аракеттерди жасайбыз:

- программанын текстин түзүүдөгүдөй жаңы терезени ачуу. Файлдын өзүнө ысмын жана кеңейтилишин беребиз.
- клавиатуранын жардамы менен текстти жазуу, ар бир сапты киргизүү `Enter` баскычын басуу менен коштолот.
- файлды төмөнкү эки жолдун бири менен сактоо:

  а) `F2` баскычын басып - эгер файлды ошол эле ат менен учурдагы каталогго, дискке сактоо керек болсо;

  б) `File` менюсун ачып, `Save As` опциясын белгилеп, `Enter` баскычын басабыз. Пайда болгон терезеден файлдын жаңы атын жана ага баруучу жолду көрсөтөбүз.

Тексттик файлдын саптарын бирден символ менен же дароо эле сап боюнча окуса болот. Мурда түзүлгөн ncg.txt файлынын мисалында мүмкүн болгон учурларды карайлы

А. Сапты символдор боюнча окуйлу:

```
program read_txt;
var f:text; s:char;
begin
assign(f,'ncg.txt'); reset(f);
while not eof(f) do begin read(f,s); write(s); end;
close(f);
end.
```

Б. Ошол эле маселенин программасын boolean логикалык функциясынын жардамы менен түзүп көрөлү:

```
program read_txt;
var f:text; ss:char;
begin
assign(f,'ncg.txt');reset(f);
while not eof(f) do begin
   while not eoln(f) do begin
     read(f,ss);write(f,ss);end;
 readln(f); {файлдагы кийинки сапка өтүү}
 writeln; {экрандагы кийинки сапка өтүү}
 end; close(f);
end.
```

II. Сапты толук бойдон окуу мүмкүнчүлүгүн берген программаны карайлы:

```
program read_txt;
var f:text; st:string;
begin
assign(f,'ncg.txt'); reset(f);
while not eof(f) do begin
   while not eoln(f) do
   readln(f,st); {файлдын сабын окуу}
   writeln(st); {аны экранга чыгаруу}
end; close(f);
end.
```

ТИПТЕШКЕН ЭМЕС ФАЙЛДАР

Типтешкен эмес файлдар мындайча баяндалат: `var g:file;` Булар үчүн компонентанын тиби көрсөтүлгөн эмес, бул болсо диск менен оперативдик эстин маалымат алмашуусун жогорку ылдамдыкта уюштурууга өбөлгө түзөт.

I. Типтешкен эмес файлдарды түзүү.

- файлды жаздыруу үчүн ачуу: `rewrite(g,n);`

n – жаздыруунун узундугу, анын мааниси 65525 тен(word) чоң болбошу керек.

- `blokwrite(<ф.о.>,<буфер>,n,k);`

мында ф.о. – файлдык өзгөрмөнүн ысмы, буфер – файлга коюлуучу өзгөрмөнүн ысмы, n – дискке бир жолу болгон кайрылуудагы жаздыруулардын саны, k – милдеттүү эмес параметр, процедурадан чыгуу мезгилинде иштетилген жаздыруулардын саны ушул параметрде кармалат

- файлды жабуу: `close(g);`

II. Типтешкен эмес файлдарды окуу.

- файлды окуу үчүн ачуу: `reset(g,n);`

n – жаздыруу узундугу, аны эске албай койсок да болот

- жаздыруунu окуу: `blokwrite(<ф.о.>,<буфер>,n,k);`

мында ф.о. – файлдык өзгөрмөнүн ысмы, буфер - дисктен окулган маалыматтар, жайгашуучу өзгөрмөлөрдүн ысмы, n – дискке бир жолу болгон кайрылуудагы окулган жаздыруулардын саны, k – милдеттүү эмес параметр, процедурадан чыгуу мезгилинде ага иштетилген жаздыруулардын саны менчиктелет

5. CRT МОДУЛУ

МОДУЛДАР

Эгер программа чоң көлөмгө ээ жана аны жөндөөдө татаалдыктар пайда болсо, эгерде сиз колдонуу мүмкүнчүлүгү кенен болгон өзүмдүк процедуралар жана функциялардын көптөгөн санын түзсөңүз, анда камтылган программаларды өзүнчө модулга (Unit) топтоштуруп койсок болот. Модулдун тексти – бул кеңейтилиши pas болгон өзүнчө турган файл, аны өзүнчө компиляциялоого болот. Эгерде модуль реалдык режим үчүн компиляцияланылса, анда компиляциянын жыйынтыгы tpu(Turbo Pascal Unit) кеңейтилишине ээ болот, корголгон режимдер үчүн компиляцияланган модулдар tpp (Turbo Pascal Protected Unit) кеңейтилишинде колдонулат. Модулдар Паскаль тилинде кеңири мааниге ээ. Тилдин стандарттык процедуралары жана функциялары алардын темасына ылайык бир нече модулдарга топтолушкан.

Модулдун түзүлүшү кадимки программанын түзүлүшүнөн айырмаланып турат. Модуль эки чоң бөлүккө ээ. Биринчиси интерфейстик, ал interface жардамчы сөзү менен башталат жана анын ичинде процедуралардын, функциялардын, константалардын жана маалыматтардын тибинин өзгөрмөлөрүн баяндоо жайгашат. Булар берилген модулду колдонуучу башка программалык модулдар үчүн да колдонулуу мүмкүнчүлүгүнө ээ болушу керек. Камтылган программалардын тексти жана берилген модулдун ичинде колдонулуу мүмкүнчүлүгүнө ээ болушкан локалдуу өзгөрмөлөр implementation (колдонуу) жардамчы сөзү менен бөлүнгөн программанын бөлүгүнөн орун алган.

Модуль Unit жардамчы сөзү жана анын ысмы менен коштолгон аталыш аймагы менен башталат. Модулдун ысмы милдеттүү түрдө ал өзү жайгашкан файлдын ысмы менен дал келиши керек.

Модулду программада же башка программалык модулда колдонуу үчүн атайын көрсөтүлгөн компилятор Uses тин жардамы менен пайдаланабыз. Uses тен кийин колдонуу модулдардын тизмеси үтүр аркылуу ажыратылып берилет.

Паскалдын библиотекалык модулдарынын ичинен, DOS аракетер системасында иштоодо төмөнкүлөр абдан чоң мааниге ээ болушат:

SYSTEM — стандарттык процедуралар жана функциялар. Бул модулдун өзгөчөлүгү ал каалаган программага кошуп койсок болот, ошондой эле аны Uses директивасы менен кошуу талап кылынбайт.

DOS — DOS аракеттер системасы менен өз ара аракет алмашуу.

CRT – экран, клавиатура жана үн динамиги менен тексттик режимде иштөө.

GRAPH – экран менен графикалык режимде иштөө.

PRINTER – принтер менен иштөө.

STRINGS – нол менен бүткөн саптар менен иштөө.

CRT МОДУЛУ

CRT модулунда экран, клавиатура жана үн динамигин башкаруучу процедуралар жана функциялар чогулган. Клавиатура менен иштөө төмөнкүдөй жол менен жүргүзүлөт. Каалаган баскычты басууда, DOS ту скан-код деп аталган, клавиатура тарабынан берилген сигналдарды иштетүүгө кошуучу аппараттык үзүлүү келип чыгат. Скан-коддор баскычтарды басуу менен, ошондой эле баскычтарды кое бергенде берилет. DOS скан-коддор-ду иштетүү менен аларды клавиатуранын атайын буферине жайгаштырат. Бул иштетүүнүн жыйынтыгы баскычтардын коду түрүндө берилет. Көпчүлүк учурда клавиатуранын скан-коддорун иштетүү талап кылынбайт жана клавиатуранын буфериндеги символдорду иштетүү менен чектелүүгө болот.

Shift жана CapsLock баскычтарынын регистрлерин эске алуу менен тамгалык-сандык баскычтарга туура келген коддор ASCII коддоруна туура келет. Ошондой эле клавиатуранын буферинен тиешелүү кодду алуу үчүн Alt баскычын басып, символдун тиешелүү сандык кодун терүү керек. Мына ушундай жол менен клавиатурада жайгашпаган псевдографикалар атайын символдор менен берилет.

Клавиатуранын буфери кезектешүү принцибине ылайыкташтырылган: жаңы келген символдор кезектин аягына жайгашат, ал эми кезектин башындагы символдор тандалат. Кезектин узундугу 15 символ менен гана чектелген. Эгер кандайдыр бир баскыч басылса, анда аргументке ээ эмес KeyPressed логикалык функциясы True (чын) маанисин берет. KeyPressed логикалык функциясы клавиатуранын буфериндеги символдордун бар же жок экендигин текшерет. Муну менен ал кезектешүү абалын өзгөртпөйт.

ReadKey функциясы Char тибине ээ жана клавиатуранын буфериндеги 1-символду кайтарат, бул учурдагы буфердеги символдун орду бошойт жана кезектин өлчөмү кичирейет. Эгерде кезек бош болсо, анда программанын аткарылышы каалаган баскычты басууга чейин токтоп турат.

KeyPressed процедурасы программаларда кандайдыр бир баскычты басуу фактысын текшерүү үчүн колдонулат. Мисалы, repeat же while циклин токтотуу үчүн:

```
Uses crt;
begin
repeat
write('мунун аяры жок');
until KeyPressed;
end.
```

```
Uses crt;
begin
while not KeyPressed do
write('мунун аяры жок');
end.
```

Бул программалар кандайдыр баскычты басканга чейин текстти экранда чыгарып турат.

KeyPressed жана ReadKey функциялары бири-бири менен байланыша алат: `if KeyPressed than key:=ReadKey;`

Эгер кандайдыр бир баскычты бассак, анда key өзгөрмөсүнө буфердеги символдун мааниси ыйгарылат. Эгер буфер бош жана баскыч басылбаса, программанын кийинки оператору аткарылат.

Программанын ишин улантуу мүмкүнчүлүгүн аныктоочу символду клавиатурадан киргизүү талап кылынган учурда ReadKey функциясын колдонуу ыңгайлуу.

Мисалы:

```
wtite('уланталыбы? (Y/N)');
repeat
    key:= ReadKey;
until key in['y','Y','n','N'];
case key of
    'y','Y': ... {ишти улантуу};
    'n','N': ... {ишти аяктоо};
end;
```

Бул жерде 'y' жана 'n' баскычтарынан башка баскычтарды басууда repeat цикли кайталана берет, качан гана бул баскычтардын бири басылганда цикл аяктайт да, программа ишин андан ары улантат.

ЭКРАН МЕНЕН ИШТӨӨ. ТЕКСТТИК РЕЖИМДЕР

CRT модулунда келтиилген маалыматты экранга чыгаруу процедуралары иштин тексттик режимдери үчүн гана түзүлгөн. Тексттик режимде экрандын талаасы саптарга жана мамычаларга болүнгөн жана талаанын ар бир позициясына ASCII коддор таблицасындагы бир гана символ чыгарылат.

Жаңы тексттик режимди орнотуу үчүн TextMode процедурасы колдонулат. Бул жерде Mode бүтүн аргументи тексттик режимдердин бирин көрсөтөт. Орнотулган тексттик режимдин маанисин CRT модулунда жарыяланган LastMode тибиндеги өзгөрмөнүн жардамы менен алууга болот.

Биз кеңири таралган режим менен иштейбиз. Текст 80 мамычадан жана 25 саптан турат, ал C80=3 константасы менен анык-

тилат. 80 мамыча жана 50 саптан турган жана кичинекей шрифтүү режимди төмөнкү константалардын суммасынан көрсөк болот: C80+Font8x8=3+256=259

Жаңы тексттик режимди орнотуу жана курсорду экрандын жогорку сол бурчуна алып келет. Экранды тазалоо үчүн ClrScr (Clear Screen – экранды тазалоо) процедурасы кызмат кылат. Төмөнкү программа C80 режими менен кичине шрифтер режиминин ортосундагы айырманы көрсөтөт:

```
program Demo;{тексттик режимди демонстрациялоо}
Uses crt;
var Mode:word;{учурдагы тексттик режимди сактоо үчүн
колдонулган өзгөрмө}
    i:byte;
begin {кеңири таралган тексттик режимди орнотуу}
TextMode(C80);
Mode:=LastMode;{учурдагы тексттик режимди сактоо}
for i:=1 to 100 do  {маалыматты 100 жолу чыгаруу}
  write('Бул C80 тексттик режими');
ReadKey; {каалаган баскычты күтүү}
{кичине бийиктиктеги тамгалар режимин орнотуу}
 TextMode(C80+256);
 for i:=1 to 100 do
  write('Бул C80+256 тексттик режими');
  ReadKey; {каалаган баскычты күтүү}
TextMode(Mode);{Негизги C80 режимин калыбына келтирүү}
end.
```

CRT модулунда экранга маалыматты чыгаруу үчүн жөнөкөй процедуралар Write жана Writeln колдонулат. Экранга кезектеги маалыматты чыгаруу экрандагы курсордун учурдагы позициясынан башталат. Мамычалар жана саптар номерленген, курсор жайгашкан мамычанын номерин WhereX функциясы менен, саптын номерин WhereY функциясы менен алууга болот. Демек x,y бүтүн сандарынын жубу монитордун экранындагы кур-сордун координатасын көрсөтөт. X огу – горизонталдуу, солдон оңду көздөй, ал эми Y огу – вертикалдуу, жогортон төмөн көздөй багытталган. Жогорку сол бурчунун координаталары (1;1) болуп эсептелет. Курсор координатасы (x,y) болгон позицияга GoToXY(x,y) процедурасынын жардамы менен жылат.

ТЕРЕЗЕЛЕР. ТҮСТӨРДҮ, ДИНАМИКТИН ҮНҮН БАШКАРУУ

Көпчүлүк тиркемелерде текстти экрандын кандайдыр тик бурчтуу областында – терезеде чыгарууга туура келет. Бул учурда экрандын четинен баштап координаталарды эсептөө кыйынчылыктарды жаратат. Мындай меселени чечүү `Window(x1,y1,x2,y2)` процедурасынын жардамы менен жүргүзүлөт. Мында бүтүн (x1,y1) сандары – экрандын жогорку сол бурчунун координаталары, (x2,y2) – төмөнкү оң бурчунун координаталары. Орнотулган учурдагы терезеде координаталар анын жогорку сол бурчунан башталып эсептелет. `ClrSqr` процедурасы да экранды учурдагы терезенин чегинде тазалайт. Терезени өчүрүү процедурасы каралган эмес. Учурдагы терезе жаңысын киргизүү менен өчүрүлөт. Жалпы экрандын чегинде иштөөгө кайтып келүү үчүн `Window(1,1,80,25)` командасы берилет.

Түстөрдү башкаруу.

Символдордун түсүн жана фонун орнотуу `TextColor, TextBackGround, LowVideo, HighVideo, NormalVideo` процедураларынын жардамы менен аткарууга болот. Стилди жана текстти экрандын ылайык келген позициясына орноткондон кийинки символдордун түсүн жана фонун өзгөртүү үчүн бул процедуралар колдонулат. Символдордун түсүн жана фонун жаңы өзгөртүү мурунку киргизилген текстке таасирин тийгизбейт. Тексттик режимдерде символдордун түсү жана фону төмөнкү константалар менен аныкталат:

Black(кара)	0	DarkGray (бозомтук)	8
Blue(көк)	1	LightBlue (көгүш)	9
Green(жашыл)	2	LightGreen (ачык жашыл)	10
Cyan(Циан)	3	LightCyan (ачык циан)	11
Red(кызыл)	4	LightRed (ачык-кызыл)	12
Magenta(сыя көк)	5	LightMagenta (ачык-сыя көк)	13
Brown(күрөң)	6	Yellow(сары)	14
LightGray (ачык- боз)	7	While(ак)	15

Символдордун түсү `TextColor(Color)` процедурасы менен орнотулат, аргументи катарында көрсөтүлгөн константалар-дын бири же word тибиндеги өзгөрмө алынат. Эгерде түстүн константасына `Blink+128` константасын кошуп койсок, чыгарылуучу символдор жаркырап, өчүп, күйүп көрүнөт. Фондун түсү 0 дөн 7 ге чейинки маа-

ишлер менен чектелет. Ал TextBackGround(Color) процедурасы менен берилет.

LowVideo процедурасы TextColor процедурасында көрсөтүлгөн символдорго салыштырмалуу алардын түсүн бозомтук кылып өзгөртөт, б.а. эгерде символдордун орнотулган түсү 8 ден 15 ке чейин болсо, анда алардан 8 кемитилет. HighVideo процедурасы буга тескери аракетти жасайт, б.а. 0..7 диапозонунда берилген түстөрдү ачык түскө өзгөртөт же 8 ге чоцойтот. NormVideo процедурасы текстик режимди орнотуудагы берилген түстү калыбына келтирет. LowVideo, HighVideo, NormVideo процедуралары кезектеги TextColor процедурасы баяндалганга чейин аракетте болушат. Качан гана бул процедура баяндалганда бул процедуралар аракетин токтотушат.

Мисалы: экрандын дал ортосунда жашыл фондо "УЛУТТУК КОМПЬЮТЕРДИК ГИМНАЗИЯ" деген сөз кызыл түс менен, андан кийинки сапка "КОШ КЕЛИҢИЗДЕР !!!" деген сөз көк түс менен жазылсын.

```
Program text;
Uses crt;
begin
clrscr; textbackground(2);gotoxy(26,12); textcolor(4);
write('УЛУТТУК КОМПЬЮТЕРДИК ГИМНАЗИЯ');
gotoxy(33,13);textcolor(9);write('КОШ КЕЛИҢИЗДЕР');
end.
```

Оңдоо процедуралары тексттин кандайдыр бир бөлүгүн өчүрүүдө, көчүрүүдө, өрдуна коюда пайдаланылат.

DelLine процедурасы курсор жайгашкан тексттин сабын өчүрөт. InsLine процедурасы курсор жайгашкан саптын астына бош сапты коет. ClrEol процедурасы курсордун учурдагы позициясынан баштап саптын аягына чейинки бардык символдорду өчүрөт.

CRT модулунда орнотулган динамикти башкаруу үчүн төмөнкү процедуралар колдонулат:

Sound(gers) процедурасы герц менен берилген жыштыгы gers болгон динамикти ишке жүктойт. Үндүн чыгышы NoSound процедурасы аткарылгандан кийин токтойт. Үнду Delay(time) процедурасынын жардамы менен кандайдыр аныкталган time убакытка карман турса болот. Үнду өчүрүү – программанын милдеттүү бөлүгү, себеби үп программа аткарылгандан кийин да улана берет.

Delay(time) процедурасы программанын аткарылышын миллисекундалар менен берилген убакытта токтотот. Бул процедура үндү же экранга чыгуучу информацияны time кандайдыр убакытка карман туруу үчүн да колдонулат.

6. ГРАФИКА МЕНЕН ИШТӨӨ

ПАСКАЛЬ ТИЛИНИН ГРАФИКАЛЫК МҮМКҮНЧҮЛҮКТӨРҮ

Паскаль программалык чөйрөсүндө көптөгөн графикалык процедуралар жана функциялар Graph модулунда чогултулган. Графикалык функциялардын библиотекасын кошуу прогаммада Uses Graph; сабы менен берилет. Бул программанын аталышынан кийин же кошулуучу модулдардын тизмесинде жайгашкан:

```
Uses Graph, <башка модулдар>;
```

Uses бөлүгүндө көрсөтүлгөн модулдар жумушчу каталогдо же Units сабындагы Options/Directories киргизүү терезесинде көрсөтүлгөн каталогдо жайгашышы керек. Мейли, graph.tpu модулу D:\BP\UNITS камтылган каталогунда жайгашсын. Анда ушул сапты UNITS киргизүү талаасына кошуу керек. Эгер бул аракетти аткарбасак, анда программаны компиляциялоодо: Error 15:File not found(Graph.tpu) маалыматы чыгат.

Графикалык режимде программанын жана видеосистеманын өз ара аракеттенишин графикалык интерфейстер BGI (Borland Graphics Interfase) колдонулган драйверлер камсыз кылышат. Драйверлер кеңейтилиши bgi болгон файлдарда топтолгон: gga.bgi,att.bgi, egavga.bgi ж.б.

Драйверлердин ар бири бир нече графикалык режимде иштөөнү камсыз кылышат. Графикалык режимди орнотуучулар *.bgi файлдарында топтолушкан.

Паскаль тилинде компьютерди графикалык режимде кошуучу InitGraph(var gd, gm: integer; path:string) процедурасы каралган. Бул жердеги gd, gm аргументтери бүтүн типке ээ жана графикалык режимди аныкташат. Графикалык режимди жабуу үчүн CloseGraph процедурасы колдонулат.

ГРАФИКАДАГЫ АЛГАЧКЫ КАДАМДАР

Түстү орнотуу. Ecavga.bgi драйвери 16 түстү колдонууга мүмкүнчүлүк берет. Ар бир түскө код – бүтүн сан менчиктелет, алар Паскаль тилинин процедуралары жана функциялары тарабынан колдонулат. Бул коддорду эсептеп отуруу кыйын болгондуктан, алар үчүн төмөнкү таблица аныкталган:

Black(кара)	0	DarkGray (бозомтук)	8
Blue(көк)	1	LightBlue (көгүш)	9
Green(жашыл)	2	LightGreen (ачык жашыл)	10
Cyan(Циан)	3	LightCyan (ачык циан)	11
Red(кызыл)	4	LightRed (ачык-кызыл)	12
Magenta(сыя көк)	5	LightMagenta (ачык-сыя көк)	13
Brown(күрөң)	6	Yellow(сары)	14
LightGray (ачык- боз)	7	While(ак)	15

Графикалык режимдеги экранга чыгарылуучу сызык жана символдордун түсүн `SetColor(Color:word)` процедурасы менен билүүгө болот. `Color` аргументинин мааниси 0 дөн 15 ке чейинки бүтүн сан же жогоруда көрсөтүлгөн константалардын бирөөсүнүн ысмы. Эгер `SetColor` процедурасы программанын аткарылышында чакырылбаса, анда ак түс колдонулат. Бул процедураны чакыргандан кийин экранга чыгарылган сызыктар жана тексттердин түсү орнотулат. Демек, сүрөт үчүн жаңы түстү тандоонун астында ар бир жолу `SetColor` процедурасын чакыруу керек.`SetbkColor(Color:word)` процедурасы бардык экрандын фонунун түсүн орнотот. Эгер бул процедура чакырылбаса, анда экран кара түстө болот.

Чекит, сызык, айлана, тик бурчтук.

`PutPixel(x,y:integer,Color:word)` процедурасы координатасы (x,y) жана түсү `Color` болгон чекитти экранга чыгарат.

`Line(x1,y1,x2,y2:integer)` процедурасы (x1,y1) чекитинен (x2,y2) чекитине чейинки аралыкта кесиндини чиет. Кесиндинин түсү `SetColor` процедурасынын жардамы менен берилет. Эгер ал берилбесе, анда кесинди ак түстө болот.

`Circle(x,y:integer,R:word)` процедурасы борбору (x,y) болгон R радиустагы айлананы сызат.

`Bar(x1,y1,x2,y2:integer)` процедурасы боёлгон тик бурчтукту сызат. Тик бурчтуктун чектери өзгөчө сызык менен белгиленген эмес. Боёнун тиби `SetFillStyle(FillStyle, Color)` процедурасы менен аныкталат. Мында `Color` – боёнун түсүн, `FillStyle` – боёнун тибин аныктайт.

Текстти киргизүү. Экранга текстти киргизүү тексттик жана графикалык режимдерде бири-биринен айырмаланып турат. `Write` жана `Writeln` процедурасын графикалык режимде колдонууга болбойт. `OutTextxy(x,y:integer,Text:string)` процедурасынын жардамы менен графикалык режимде текстти киргизүү ишке ашырылат. Мында x,y – киргизилүүчү тексттин жогорку сол бурчунун координаталары, `Text` – берилген сап, б.а. киргизилүүчү текст.

БИРИНЧИ ГРАФИКАЛЫК ПРОГРАММА

Төмөндө келтирилген программа экранга текстти, айланаңы жана квадратты чыгарат, фондун, сызыктардын жана толтуруунун түсүн көрсөтөт. Андан соң экранга кайра текстти, айланаңы, квадратты чыгарат.

```
Program Bar;
uses graph;
var gd,gm,errorcode:integer;
begin
gd:=detect;initgraph(gd,gm,'');errorcode:=GraphResult;
if  errorcode=grOk then begin
Circle(50,50,30); {ак түстөгү айлана}
Bar(30,30,70,70); {айлананын ичиндеги ак квадрат}
Outtextxy(50,100,'улантуу үчүн Enter');readln;
setbkcolor(red); {фонду кызыл түстө орнотуу}
setcolor(yellow); {сызыктын сары түсүн орнотуу}
Setfillstyle(1,9);{областарды боё үчүн көк түстү орнотуу}
Circle(150,50,30); {сары түстөгү айлана}
Bar(130,30,170,70); {көк түстө боёлгон квадрат}
{ак түстөгү айлана жана квадраттын оң жагында сары
айланада көк квадрат тартылган}
readln;
closegraph; {графикалык режимди жабуу}
end else writeln('grafics error');
end.
```

Программадан көрүнүп тургандай алгач экрандын кара фонунда ак түстөгү айлананын ичине сызылган ак түстөгү квадрат чыгат жана Enter баскычын басууну сурайт. Андан кийин фондун түсү өзгөрөт жана көгүш квадрат, сары айлана сүрөттөлөт. Enter ди кайталап басканда графикалык режим жабылат жана программа аяктайт.

ГРАФИКАЛЫК КӨРСӨТКҮЧ

Графикалык режимде көрсөткүч түшүнүгү киргизилет. Ал текстик режимдеги курсор түшүнүгүнө окшош болот. Кээ бир процедуралар көрсөткүчтүн координатасы менен байланышкан графикалык образдарды чыгарат, башкалары абсолюттук координаталар менен иш алып барышат. Көрсөткүч, курсордон айрмаланып, көрүнбөйт.

MoveTo(x,y:integer) процедурасы көрсөткүчтү жаңы (x,y) чекитине которот. MoveRel(DeltaX, DeltaY:integer) процеду-

расы көрсөткүчтүн учурдагы позициясынан аны (DeltaX, DeltaY) аралыкка жылдырат. Кээ бир процедуралар көрсөткүч аныктаган чекиттен баштап сызыктарды чиет жана тексттерди киргизет. Мындай учурларда көрсөткүч перо сыяктуу которулуп, өзүнүн координаталарын өзгөртөт. GetX:integer жана GetY:integer функциялары көрсөткүчтүн учурдагы абалын берет.

Кээде бардык экран менен эмес, анын бөлүнгөн терезе менен иштөөгө туура келет. Паскалда терезе түшүнүгү жетишээрлик деңгээлде берилген эмес, бирок анын кээ бир окшоштуктары – чыгаруу областары визуалдуу портту (VievPort) колдонсок болот. Анын негизги багыты – сүрөттөлүштү чыгаруу үчүн тик бурчтуу активдүү областы бөлүп алуу.

SetVievPort(x1,y1,x2,y2:integer,Clip:boolean) процедурасы сүрөттөлүштү чыгаруу үчүн экрандын областын орнотот. (x1,y1) – визуалдык порттун жогорку сол бурчунун, (x2,y2) – төмөнкү оң бурчунун абсолюттук координаталары. Процедуранын аткарылышынан кийин координата системасынын башталышы порттун жогорку сол бурчуна которулат.

Эгерде сүрөттөлүш берилген тик бурчтуктун чектеринен чыгып кетсе, анда аны чыгарбоо же чыгаруу керек экендигин логикалык типтеги Clip өзгөрмөсү аныктайт. Эгер Clip=False сүрөттөлүш белгиленген чектердин сыртында да чыгарылат.

ClearVievPort процедурасы учурдагы визуалдык порттун областындагы бардык сүрөттөлүштөрдү өчүрөт. Учурдагы портту алып салуу процедурасы каралган эмес. Эгер биз толук экран менен иштөөгө кайрылгыбыз келсе, анда SetVievPort(0,0,GetMaxX, GetMaxY,True) процедурасын колдонушубуз керек.

ClearDevice процедурасы көрсөткүчтү координаталар башталышына кайтарып келет, орнотулган визуалдык порттогу алып салат жана бардык сүрөттөлүштөрдү өчүрүп, берилген фондогу түстү гана алып калат.

СЫЗЫКТАР ЖАНА ФИГУРАЛАР

Graph модулунун библиотекасында кесиндилерди, айланаларды, эллипстерди, жаалардыы, тик бурчтуктарды сызуу үчүн бир нече процедуралар каралган. Бул процедуралардын бардыгында сызыктар менен иш алып барабыз. Сызыктын алгачкы түсү ак болот. Сызыктын түсүн жана стилин өзгөртсөк болот. Сызыктын түсү SetColor процедурасы менен өзгөрүлөт. Процедура жаңы түс орнотулганга чейин, кийинки сүрөттөлүштөргө колдонула берет. Тилекке каршы, кара түс менен түстүү фондо сызык сызуу каралган эмес

Эгер `SetBkColor(Yellow);SetColor(Black);` процедураларын берсек, анда сүрөттөлүш экранда көрсөтүлбөйт.

Сызык чийилүү стили(жөнөкөй, пунктир, чекиттик, ж.б.), калыңдыгы жана боёлуу аракети менен мүнөздөлөт. Сызыктын стилинин 5 варианты каралган, аларга 0 дөн 4 кө чеңинки сандар же константалар туура келет.

`SolidLn=0` – жөнөкөй сызык

`DottedLn=1` – чекиттүү сызык

`CenterLn=2` – пунктир-чекитүү сызык

`DashedLn=3` – пунктир сызык

`UserBitLn=4` – тиби колдонуучу тарабынан аныкталган сызык.

Колдонуучу тарабынан берилген типти аныктоо үчүн боё шаблонун берүү керек(`Pattern`). Сызыктын жоондугунун эки тиби каралган: бирдик `NormWidth=1` жана үчтүк `ThickWidth=3`.

`SetLineStyle(Ls:word,Pattern:word,Width:word)` процедурасы сызыктын тибин орнотууну ишке ашырат. Процедуранын 1-аргументи жогоруда көрсөтүлгөн 5 шаблондун бирин аныктайт, акыркысы сызыктын жоондугун, экинчиси качан гана шаблон катарында `UserBitLn` берилгенде маанеге ээ болот.

`Line(x1,y1,x2,y2:integer)` процедурасы (x1,y1) чекитинен (x2,y2) чекитине чейинки кесиндини сызат, бул жерде көрсөткүчтүн абалына эч кандай таасири тийбейт.

`LineTo(x,y:integer)` процедурасы сызыкты көрсөткүчтүн учурдагы абалынан (x,y) чекитине чейин сызат, ошондой эле көрсөткүчтү ошол чекитке которот.

`LineRel(DeltaX,DeltaY:integer)` процедурасы көрсөткүчтүн учурдагы абалынан баштап, координата башталышынан `DeltaX,DeltaY`) аралыкта жаткан чекитке чейинки сызыкты сызат, көрсөткүч акыркы чекитке которулат.

`Rectangle(x1,y1,x2,y2:integer)` процедурасы бир бурчу (x1,y1) координатасы менен берилген, ал эми ага каршы бурчу (x2,y2) координатасы менен берилген тик бурчтукту сызат.

`Circle(x,y:integer,R:word)` процедурасы борбору (x,y) чекити, радиусу R болгон айлананы сызат.

`Arc(x,y:integer,StartAngle,EndAngle:word,R: word)` процедурасы борбору (x,y) чекити, радиусу R, градустар менен берилген баштапкы бурчу `StartAngle` жана акыркы бурчу `EndAngle` болгон айлананы жаасын сызат. Жаа `StartAngle` дан баштап `EndAngle` ге көздөй саат жебесине каршы чийилет. Бурчтар 0 дөн 359 га чейинки диапазондо жатышы керек, эгер бурч 359 дан чоң болсо, анда анын чондугунан айлануулардын саны (бир айлануунун саны 360° ка барабар) кемитилет.

Мисалы: Arc(100,100,0,365,50) жана Arc(100,100, 0,5,50) бир эле жыйынтыкка алын келет.

Ellipse(x,y:integer,StartAngle,EndAngle:word, Rx, Ry:word) процедурасы эллипсти же анын жаасын жазат. (x,y)– борборунун координаталары, StartAngle жана EndAngle – баштапкы жана акыркы бурчтар, алар Arc процедурасындагыдай эле аныкталат, Rx, Ry – эллипстин пикселдер менен берилген жарым окторунун радиустары.

DrawPoli(NumPoints:word, var PolyPoints) процедурасы чекиттердин массиви менен берилген сынык сызыкты сызат. NumPoints аргументи – сызыктын чекиттеринин санын берет, PolyPoints – чекиттердин, б.а. объекттердин массиви болуп эсептелет.

```
PointType=record
x,y:integer;
end.
```

Мисалы: туюк үч бурчтукту сызуу үчүн var triangle: array[1..4] of PointType өзгөрмөсү баяндалып, үч бурчтук-тун бурчтарынын координаталары баяндалышы керек: Triangle[1]. x:=... ж.б. Үч бурчтукту түзүү үчүн DrawPoly(4, triangle) процедурасын чакыруу керек.

БОЁЛГОН ФИГУРАЛАР

Боёнун типтери SetFillStyle же SetFillSettings процедуралары менен берилет. Bar(x1,y1,x2,y2:integer) процедурасы боёлгон тик бурчтукту берет. Тик бурчтуктун чектери өзгөчө сызык менен белгиленбейт.

Bar3D(x1,y1,x2,y2:integer,Depth:word,Top:boolean) процедурасы тик бурчтуу параллелепипеддин аксионометриялык сүрөттөлүшүн берет. (x1,y1) жана (x2,y2) – анын фронталдуу грандарынын координаталары, Depth – калындыгы(жоондугу). Логикалык типтеги Top параметри гранды сызуу кереги барбы же жокпу, ушуну аныктайт. Эгер Top=TopOn=True, анда жогорку гран сүрөттөлөт, эгер Top=TopOff=False, анда жогорку гран сүрөттөлбөйт, фронталдуу граны гана боёлот. Бардык чектери SetLineStyle процедурасы менен аныкталган сызыктар менен чийилет.

FillEllipse(x,y:integer,Rx,Ry:word) процедурасы борбору (x,y) чекити жана жарым октору Rx, Ry болгон боёлгон эллипсти берет. Эллипстин чеги – SetLineStyle процедурасы менен берилген стилге жараша бирдик же үчтүк калыңдыктагы бүтүн сызык.

PieSlice(x,y:integer,StartAngle,StopAngle,R:word) процедурасы борбору (x,y), радиусу R, StartAngle жана StopAngle бурчтары менен чектелген, боёлгон айлананын секторун берет. Секторду чектеп турган сызыктар SetLineStyle процедурасы менен берилет, сырткы жаа сызык менен белгиленбейт.

Sector(x,y:integer,StartAngle,Stop-Angle,Rx,Ry: word) процедурасы эллипстин боёлгон секторун берет, ага PieSlice процедурасында айтылгандардын баары тиешелүү.

FillPoly(NumPoints:word,var Poly-Points) процедурасы боёлгон көп бурчтукту берет, ал DrawPoly процедурасына аналогиялуу түрдө аныкталат.

Мисалы:

```
program Lines;
uses crt,graph;
var gd:integer;  {драйвер}
    gm:integer;  {графикалык режим}
    errcode:integer;
    LineStile:integer;{сызыктын стили}
    LineWidth:integer;{сызык. жоондугу}
const triangle:array[1..4] of
Pointtype=((X:300;Y:150),(X:350;Y:170), (X:320;Y:250),
(X:300;Y:150))
{үч бурчтуктун чокуларынын координаталары}
begin
gd:=detect;initgraph(gd,gm,'');errcode:=graphResult;
if errcode=grOk then begin
setbkcolor(lightgreen); {фондун түсү}
setcolor(lightred); {сызыктын түсү }
putpixel(20,20,white); {чекитти сызуу}
for LineStile:=-1 to 4 do
for LineWight:=1 to 3 do
if LineWight<>2 then
begin cleardevice; {экранды тазалоо}
{сызыктын стили}
setlinestyle(LineStile,0, LineWight);
{экрандын жогорку жарымына портту орнотуу}
setviewport(0,0,getmaxX, getmaxY div 2,false);
moveto(20,5);
case LineStile of
0:outtext('solidln=0'); 1: outtext(dottedln=1');
2: outtext('centerln=2'); 3: outtext('dashedln=3');
4: outtext(userbitln=4');
end;
if LineStile<> -1 then
```

```
case LineWight of
 1: outtext('normwid=1'); 3: outtext('thickwid=3');
end;
{контурдук фигуралар}
circle(100,70,50); {айлана}
line(200,40,250,50); {сызык}
moveto(270,40);{көрсөткүчтү которуу}
lineto(320,50);moverel(0,-20);linerel(50,20);
rectangle(400,30,450,70); {тик бурчтук}
arc(100,200,315,45,40); {айлананын жаасы}
ellipse(350,200,90,180,150,100); {эллипстин жаасы}
drawpoly(sizeof(triangle) div sizeof (pointtype),
triangle); {үч бурчтук}{фигураларды боё}
setvievport(0,getmaxX,getmaxY div 2, ,false);
{экрандын төмөнкү жарымына портту орнотуу}
bar(10,10,50,50); {тик бурчтук}
fillpoly(sizeoff(triangle) div sizeof (pointtype),
triangle); {үч бурчтук}
bar3d(70,30,100,100,20,topon); {параллелепипед}
pieslice(200,50,45,315,50); {айлананын сектору}
sector(350,50,45,315,50,30); {эллипстин сектору}
fillellipse(100,200,50,20); {боёлгон эллипс}
readln;end;closegraph;end
else writeln('graphics error');
end.
```

ТЕКСТТЕРДИ ЖАЙГАШТЫРУУ

Алгач текст кандай касиеттер менен мүнөздөлөт, ушуларды тактап алалы.

- жазуунун багыты(вертикалдуу же горизонталдуу)
- текст жазылуучу экрандын чекитинин координаталары
- юстировкалоо, б.а. экрандын көрсөтүлгөн координатасына карата текстти жайгаштыруу
- шрифтти сызуу
- шрифттин өлчөмдөрү

Жазуунун багыты `word` тибиндеги эки константа менен берилет:

- `HorizDir=0` – горизонталдуу багыт
- `VertDir=1` – вертикалдуу багыт

Текстерди чыгаруу үчүн `Outtext(Text:string)` жана `OuttextXY(x,y:integer,Text:string)` процедуралары колдонулат. Биринчиси текстти чыгаруунун координатасы катары учурдагы көрсөткүчтүн абалын колдонот, процедура аткарылгандан кийин

көрсөткүчтүн X координатасы чыгарылган тексттин узундугуна барабар өсүндүгө чоңоёт. Бул процедураны бир сапка бир нече жазууларды чыгарууда колдонуу ыңгайлуу. Мында көрсөткүчтүн автоматтык которулуусунун жардамы менен ар бир кийинки жазылыштын узундугунун мурункусунан болгон көз карандылыгында ар бир кийинки жазылыштын координатасы туура аныкталат.

Экинчи процедура көрсөткүчтүн координаталарына эч таасирин тийгизбестен, чыгаруу чекиттеринин координаталары так, даана түрдө берет. Бул жерде тексттердин бири-бирине кийишип калуусунан сак болуу керек.

Текстти чыгаруунун координаталары анын сол жаккы бурчунун координаталары болуп эсептелинишет. Бирок юстировалоонун башка типтерин да колдонуу мүмкүнчүлүктөрү бар. Юстировкалоо X жана Y октору боюнча көз карандысыз жүргүзүлөт. Мында тексттин эки жагын октор боюнча борборго карата жайгаштырууну да жүргүзсөк болот.

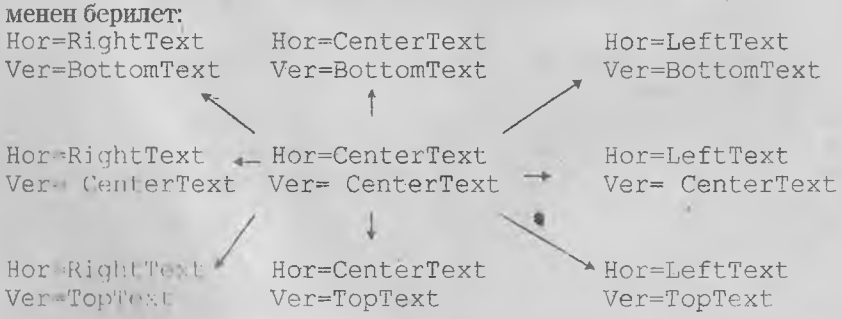
Горизонталдуу юстировкалоо:

`LeftText=0` – экрандын сол жагына карата жайгаштыруу;
`CenterText=1` – тексттин борбору боюнча жайгаштыруу;
`RightText=2` – экрандын оң жагына карата жайгаштыруу.

Вертикалдуу юстировкалоо:

`BottomText=0` – экрандын ылдыйкы жагына карата жайгаштыруу;
`CenterText=1` – борбор боюнча жайгаштыруу;
`TopText=2` – экрандын жогору жагына карата жайгаштыруу.

Текстти борборго карата жылдыруу багыты төмөнкү диаграмма менен берилет:



`SetTextJustify(Hor,Ver:word)` процедурасы аркылуу юстировкалоонун тибин орнотуу жүргүзүлөт. Мында `Hor` жана `Ver` – горизонталдык жана вертикалдык багытта юстировкалоонун тиби. Мисалы:

```
SetTextJustify(CenterText, CenterText);
OutTextXY(getmaxX div 2,getmaxY div 2, Text);
```

кайрылуусу Text өзгормөсүнө менчиктелген сапчаны экрандын борборуна чыгарат.

ШРИФТТЕР

Колдонуучу бир нече шрифттер менен иштей алат, ошондой эле символдордун өлчөмдөрүн жана түстөрүн өзгөртүүгө мүмкүнчүлүгү бар. Жазуулардын түсү SetColor процедурасы менен аныкталат.

Шрифттердин көрсөтмөлүүлүгүн эки түрдө айырмаласак болот: матрицалык жана масштабдык. Матрицалык шрифтерде символдор өлчөмү 8x8 чекит болгон мозаикалык сүрөттөр болуп эсептелишет. Шрифтерди бир нече өзгөртүүдө ар бир чекит тиешелүү өлчөмдүн квадратына трансформирленишет, бул саптык эмес сызылуу катары кабыл алынат. Ошондуктан матрицалык шрифттер жөнөкөй тексттик информацияны чоңойтуусуз чыгаруу үчүн колдонулат. Матрицалык шрифтердин жакшы жагы – чыгаруу тездиги жана символдорду сактоодо аз өлчөмдөгү эске ээ болушу.

Жогорулатылган талаптарда тамгаларды чийүүдө масштабдык шрифтер, б.а. өлчөмүн чоңойткон мезгилде сапатын жоготпогон шрифттер колдонулат. Бул шрифттер жөнүндөгү информация кеңейтилиши chr болгон атайын файлдарда сакталат. Винчестрде(сырткы эс) жайгашкан шрифттерге кайрылуу бир топ убакытты талап кылат. Мындан сырткары программаны колдонууда учурдагы каталогдо шрифттин файлынын болушун камсыз кылуу керек.

Биз бир матрицалык жана бир нече масштабдык шрифтерди карайлы. Колдонуучу шрифтти көрсөтүү үчүн аларга туура келген константаларды колдонобуз.

DefaultFont=0	**өлчөмү 8x8 болгон матрицалык шрифт**
TriplexFont=1	**“Триплех” масштабдык шрифти**
SmailFont=2	**майда өлчөмдөгү масштабдык шрифт**
SansSeriftFont=3	**“Сан-Сериф” масштабдык шрифти**
GothicFont=4	**Готикалык масштабдык шрифти**

Учурдагы шрифтти орнотуу үчүн SetTextStyle(F:word, Dir:word,Size:word) процедурасы колдонулат. Мында F – шрифттин тиби, Dir – жазуунун вертикалдык же горизонталдык багытын көрсөтүү белгиси, Size – өлчөм, бүтүн коэффициент, ага тамгалардын өлчөмү көбөйтүлөт.

Кээ бир учурларда Size өзгормөсү каалаган эффектке жетүү үчүн жетишсиз болуп калат. Масштабдоо мүмкүнчүлүктөрүн кеңейтүү үчүн SetUserSize процедурасы колдонулат. Ал X огу боюнча даяр Y

огу боюнча дагы шрифттин өлчөмдүү масштабын орнотууга мүмкүндүк берет. Бул процедуранын аргументтери болуп бүтүн сандар эсептелинет. X жана Y октору боюнча шрифттин өлчөмүн бул сандарга көбөйтүп жана бөлүп ала алабыз. Демек, масштабдуу произволдуу коэффициент рационалдык бөлчөктүн алымы жана бөлүмү түрүндө берилет. Бул процедуранын жыйынтыгын колдонуу үчүн `SetTextStyle` процедурасындагы `Size` параметринин маанисин нөлгө барабарлап алышыбыз керек.

Мисал: Системада орнотулган шрифттерди демонстрациялоочу программаны карайлы. Шрифттердин тиби ар бир жолу `Enter` баскычын басканда өзгөрүп турат. Системада жогоруда көрсөтүлгөн 5 шрифттен (0 дөн 4 кө чейинки номердеги) башка константанын аты менен эмес, номер менен орнотулган шрифттер да болушу мүмкүн.

Көңүл бургула! Шрифттин номерин аныктоодо системада каралганга караганда чоң болгон `DefaultFont` матрицалык шрифти колдонулат, бирок өлчөм масштабдалбайт.

```
Program TextStyles;
uses graph;
var errorcode,gd,gm,y,s,ypos:integer;
    Font:word;FontName:string;
procedure GFN(var FontName:string);
{процедура орнотулган шрифтин константасынын атын
кайтарып берет}
var TextSettings: TextSettingsType;
begin
  {шрифтти орнотуу аракеттерин аныктоо}
  GetTextSettings(TextSettings);
  case TextSettings.Font of
  0:FontName:='DefaultFont';
  1:FontName:='TriplexFont';
  2:FontName:='SmallFont';
  3:FontName:='SansSeriftFont';
  4:FontName:='GothicFont';
  else FontName:='Name unknown';end
end;
begin
 gd:=detect; initgraph(gd,gm,'');
 if graphresult<>grOk then Halt(1);
 SetTextJustify(CenterText,CenterText);
 for Font:=0 to 10 do begin ClearDevice;
 {кириизүү чегитинин вертикалдуу координатасынын
 баштапкы маанисин орнотуу}
 y:=0; {y координатасы жазуунун бийиктигине барабар
  узундукка чоңоет}
```

```
  inc(y,TextHeight('Size')+1);
  {шрифттин бийиктиги боюнча жыйноо}
  for S:=1 to 5 do begin
  SetTextStyle(Font,HorizDir,s); {шрифтти орнотуу}
  if effofcode<>0 then
  OutTextXY(getmaxX div 2,y, 'Error');
  else
  OutTextXY(getmaxX div 2,y,'Size='+ Chr(s+48));
  inc(y,TextHeight('Size'+1);
  end;
 {шрифттин аты жөнүндөгү стилди орнотуу}
 SetTextStyle(Font,HorizDir,2);
 GFN(FontName); {шрифттин атын аныктоо}
 if FontName='Name unknown' then
 OutTextXY(getmaxX div 2, y,FontName);
 else
 OutTextXY(getmaxX div 2, y,'It is  '+  FontName);
 readln;
 end;
closegraph;
end.
```

III бөлүм
QBASIC
ПРОГРАММАЛОО ТИЛИ

1. БЕЙСИКТИН ПРОГРАММАЛЫК ЧӨЙРӨСҮ

АЛГАЧКЫ КАДАМ

Компьютерде маселелерди чыгарууда логикалык, математикалык эсептөөлөр, сүрөттөлүштөр аткарылат. Бул бөлүм көпчүлүккө таанымал болгон Qbasic программалык тилине арналат. Qbasic(Beginner All-purpose Symbolic Instruction Code)– программа түзүүнү жаңыдан баштагандар үчүн ASCII коддорунда символдор менен берилген универсалдуу тил болуп сана-лат. Анын алгачкы версиясы Basic 1965-жылы АКШдагы Дартмут кол-леджинин профессорлору Т. Куртц жана Дж. Кеменилер тарабынан эсептөө техникасы менен тааныштыгы жок студенттерди окутуу үчүн түзүлгөн. Кийинчерээк Basicти өзгөртүп түзүү менен анын көптөгөн версиялары пайда болду. Азыркы учурдагы негизги системалар CW-Basic,Turbo Basic,Visual Basic,Qbasic ж.б. колдонулууда.

Qbasicти иштетүү үчүн активдүү дисктеги Бейсиктин файлдары жайгашкан каталогдогу qbasic.exe файлын жүктөө жетиштүү. Мисалы: биздин каталог QB болсо, анда DOS тун командалык сапчасында төмөнкүдөй көрүнүштө орун алат:

```
C:\QB
C:\QB\qbasic.exe
```

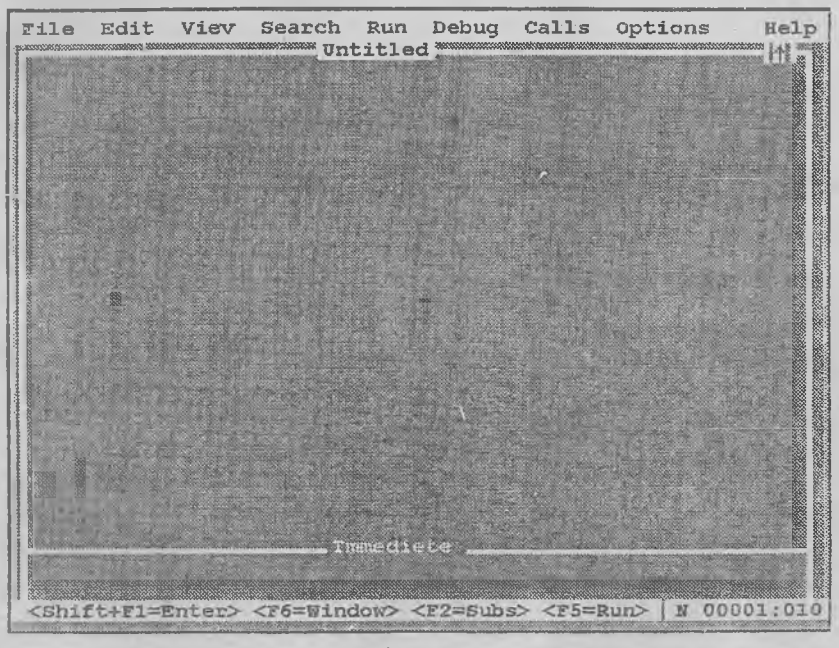
Qbasicтин жалпы көрүнүшү 4-сүрөттө көрсөтүлгөн. Мындан ары бизге ыңгайлуу болсун үчүн Qbasicти Бейсик деп атайбыз.

Көпчүлүк учурда Бейсик тилиннин негизи системалык программалоо, интегралдоо каражаты катарында түзүлөт жана төмөнкүлөрдү өзүнө камтып турат:

- файлдык система;
- компилятор;
- жөндөө;
- экрандык редактор;
- байланыш редактору;
- ички маалымат системасы

Бейсиктити иштетүү менен экранда төмөндөгүлөрдү көрүүгө болот.

- ар кандай аракеттерди көрсөтүүчү негизги меню;
- төмөнкү сапта негизги клавишалар боюнча көрсөтмө;
- экрандын негизги бөлүгүн ээлеп, программанын текстин көрүү, өзгөртүү аракеттери аткарылуучу редактор терезечеси жайгашкан.



4-сүрөт

Бейсик программалоо тилинин негизги менюсу төмөнкүдөй пункттар менен коштолгон:

File – файлдар менен иштөө

Edit – текстти редактирлөө, анын бөлүктөрүн көчүрүү жана жылдыруу

View – көрүү

Search – керектүү текстти издөө, сөздөрдү алмаштыруу, издөөнүн башка инструменттери

Run – программаны аткаруу, кадам боюнча жөндөө

Debug – жөндөө программасынын инструменттери

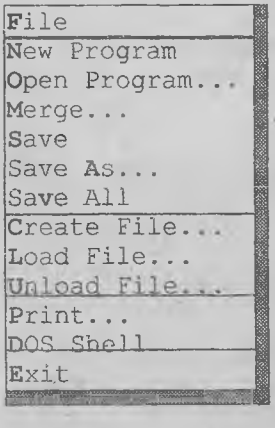
Option – программа чөйрөсүн жөнгө салуу

Help – жардам

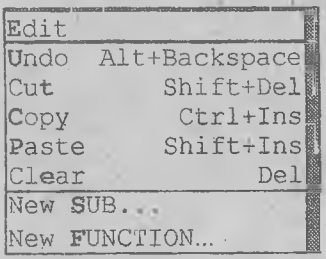
Алардын кээ бирлеринин аткарган кызматын карап чыгалы.

File менюсунун аткарган кызматы:

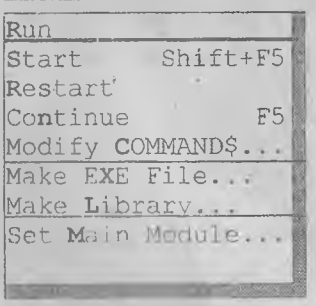
File менюсунун аткарган кызматы:



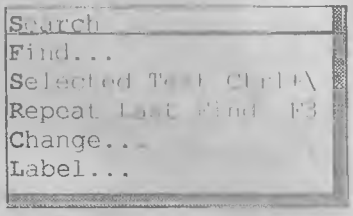
New Program – жаңы файл түзүү
Open Program...– мурда түзүлгөн файлды ачуу
Merge...– бириктирүү
Save – түзүлгөн файлды сактоо
Save As...– файлдын атын жана багытын көрсөтүп дискке сактоо
Save all – бардык файлдарды сактоо
Create File...– файл түзүү
Load File...– жүктөө
Unload File...– жабуу
Print...– файлды кагазга чыгаруу
DOS Shell – DOS тун терезеси
Exit – Бейсик чөйрөсүнөн чыгуу



Edit менюсунун аткарган кызматы:
Undo – акыркы аракетти кайтаруу
Cut – буферге жайгаштыруу
Copy – буферге көчүрүү
Paste– буфердеги текстти жайгаштыруу
Clear – өчүрүү
New SUB...– жаңы программа
New FUNCTION...– жаңы функция

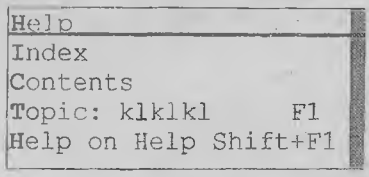


Run менюсунун аткарган кызматы:
Start – программаны аткаруу
Restart – кайрадан баштоо
Continue – улантуу
Modify COMMAND$...– DOSтун командалык сапчасы менен болгон байланыш
Make EXE File...– файлды .EXE файлга которуу
Make Library...– файлды объективдүү файлга которуу
Set Main Module...– негизги менюну тандоо



Search менюсунун аткарган кызматы:
Find...– көрсөтүлгөн символду издөө
Selected Text – текстти тандоо
Repeat Last Find – кайталоо
Change...– алмаштыруу
Label...– метка

Help **менюсунун** аткарган кызматы:



Index – командаңын баштапкы тамгасы аркылуу жардам
Contents – системалык чөйрө боюнча жардам
Topic: klklkl – учурдагы команда тууралуу жардам
Help on Help - Help менюсу менен иштөөдөгү көрсөтмөлөр

БЕЙСИКТИН НЕГИЗГИ ЭЛЕМЕНТТЕРИ

Бейсиктин негизги элементтери болуп маалыматтар, туюнтма, операциялар(амалдар) жана түшүндүрмөлөр эсептелет. Бейсикте программа түзүүдө төмөндөгү негизги символдор колдонулат. Латын тамгалары(A-Z, a-z), Араб цифралары(0-9), атайын символдор колдонулат. Бейсикте маалыматтар сандык, саптык жана массивдер болуп бөлүнөт. Саптык(литердик) жана сандык чоңдуктар өз ара өтүндө өзгөрмөлүү жана туракту болуп экиге бөлүнөт. Туракту чоңдуктар жана түшүндүрмөлөрдү жазууда орус тамгалары колдонулат (А-Я, а-я). Чоңдуктардын үстүнөн саптык, арифметикалык, логикалык, катыш амалдары аткарылат. Амалдарды аткарууда бекитилген кээ бир символдорго токтололу.

Математикалык операторлор			
*	көбөйтүү	<, >	салыштыруу белгилери
–	кемитүү	+	кошуу
/	бөлүү(слэш)	.	ондук үтүр
^	даража белгиси	\	бүтүн сандык бөлүү белгиси
=	барабардык же менчиктөө символу		

Маалыматтардын типтерин белгилөөдө төмөндөгүдөй белгилер колдонулат:

!	жөнөкөй тактык	$	саптык чоңдук	%	бүтүн
#	экилик тактык	&	узун бүтүн		

Бейсикте программалоо тилинде колдонулуучу атайы символдор:

, ;	PRINT жана INPUT операторлорунун форматында колдонулуучу символдор
:	бир сапта берилген операторлорду бөлүп турган символу
‘	түшүндүрмөлөр сапчасы(апостроф белгиси)
?	INPUT операторунун чакыруусу

Программанын синтаксистик элементери болуп өзгөрмөлөрдүн аталышы, кызматчы сөздөр, туракттуулар, чектөөлөр эсептелет

ЧОҢДУКТАР ТҮШҮНҮГҮ. ТУРАКТУУ ЧОҢДУКТАР

Программанын иштеши менен мааниси аныкталган жана өзгөрбөгөн чоңдуктар туракту чоңдуктар болуп эсептелет. Бейсикте бүтүн, аныкжана символдук туракту чоңдуктар колдонулат. Бүтүн турактуулар бүтүн сан түрүндө жазылат. Бейсикте –32768 ден 32767 чейинки сандар колдонулат. Бүтүн турактууларга мисал келтирели: –15; 47; 0; –513; 641, ж.б. Туура эмес жазылган турактуулар: 46,7; 167.; 895468123

Жогорку мисалдарда көрсөтүлгөндөй ондук чекит коюу, үтүр белгиси(бөлчөк сан), максималдык санды берүү туура эмес.

Аныктуракту уулар. Бейсикте 10^{-38}нан 10^{+38} чейинки диапозондогу аныксандар колдонулат. Эгер программанын иштешинде эсептөөлөр бул диапозондон чыгып кетсе, анда программа иштебей токтоп калат ошондой эле бул тууралуу билдирүү берет.

Бейсикте аныксандардын жазылышына мисал келтирели:

Сан	Бейсик тилинде	Сан	Бейсик тилинде
0,56	0.56 же жөн эле .56	- 45,654	- 45.654
- 89,0	- 89.0 же – 89	9,6	9.6

Өтө чоң же өтө кичине аныксандар менен иштөөдө, сандардын бүтүн жана бөлчөк бөлүгүн бөлүп көрсөтүү үчүн E символун колдонуу жазылат. Аныксандарды бул түрдө жазууда E символу 10 негизи катары пайдаланылат. Ал сандын көрсөткүчү E символунан кийин белгиси көрсөтүлүп бүтүн константалар түрүндө көрсөтүлөт. Мисалы:

Сан	Бейсик тилинде	Сан	Бейсик тилинде
10^{15}	1.0E15 же 1E15	– 0,000059	$-5.9*10^{-5}$ же – 5.9E-5
– 54100	$-541*10^{2}$ же – 541E2	1,635	0,1635*10 же 0.1635E1

Бул мисалдарда байкагандай аныксандардын маани берүүчү цифралары алтыдан ашпашы керек. Аныкконстанталарды бүтүн константа түрүндө ондук көрсөткүчү аркылуу өткөрүүгө токтолдук. Ошентип аныкконстаналар жазылышы боюнча кстирилүүчү каталарга мисал келтирели:

9,7 – үтүр ордуна чекит(.) болуусу керек
0 – ондук корсөткүч же ондук чекит жок болуусу
5.4E2.4 – E нин даражасы ондук көрсөткүч аркылуу берилген
87E-E – E символунан кийин бүтүн константа турууга тийиш
7,54E145 – ондук корсөткүч максималдык мааниден чоң сан болуп калган

Символдук(литердик) турактуулар.

Бейсик тилинде символдук чоңдуктар да колдонулаарын да билебиз. Символдук турактуу деп Бейсик тилинин алфавиттик символдорунун тамга, цифра, атайын символдор тырмакчага алынган каала-

гандай тобун айтабыз. Символдук туракгуулар тырмакча менен белгиленип жазылат. Мисалы: "Алина" , " Бишкек–2001", " 45 + 87 ", "А В С "

ӨЗГӨРМӨ ЧОҢДУКТАР

Бейсик тилинде маселенин чыгарылышына программа түзүүдө ячейкалардын абсолюттук адрестеринин ордуна ячейкалардын бир же бир нече символ менен белгиленген белгилери жана сандар колдонулат.

Өзгөрмө – эстин кандайдыр бир бөлүгүн ээлеген жана кандайдыр бир мааниге ээ болгон чоңдуктун символдук көрсөтүлүшү. Өзгөрмөнүн аталышы деп өзгөрмөлөрдү белгилөөдө колдонулган символдордун тобу аталат. Өзгөрмөлөрдү белгилөө тамга менен башталып, тамга, цифралардан турат жана жогоруда көрсөтүлгөн символдор менен аныкталып, тиби аныкталат. Өзгөрмөнүн аталышы, анын ысымы же **идентификатор** деп аталат.

Компьютердин оперативдик эси бөлүкчөлөрдүн чогуусунан тураарын жана ар бир бөлүкчө бир санды, символду же бир команданы эстей ала турганын билебиз. Программанын аткарылышы менен маанилерин өзгорткөн чоңдуктар **өзгөрүлмөлүү чоңдуктар** болуп эсептелет.

Символдук өзгөрмөлөр болсо печатка тырмакчасыз берилет. Программанын иштеши менен компьютердин эсиндеги ар бир өзгөрүлмө(сандык же символдук) эстин бөлүкчөлөрүнөн (сакталган ордунан) алынат, анын ордуна кийинки өзгөрүлмөнүн мааниси келип түшөт. Бул өзгөрүлмөнүн мааниси ячейканын жаңы өзгөрүлмөсү менен окулат. Өзгөрулмөнүн берилишине мисал келтирели: `X, XY, A1, P, KL, SUMMA`

Өзгөрмөнүн туура эмес жазылышына мисал келтирели:

`5X` – цифра биринчи орунда жазылган;

`K*` – "*" символун колдонууга болбойт;

Бейсикте өзгөрүлмөлөр төмөндөгү үч типте болот:

1. Бүтүн озгормө 2. Анык өзгөрмө 3. Символдук өзгөрмө

1. Бүтүн өзгөрмөнү көрсөтүү үчүн `%` белгиси колдонулат жана ал өзгөрмөнүн ысмынан кийин коюлат. Мисалы: `K%, AB%, N5%`. Программа аткарылып жатканда бүтүн өзгөрмөлөрдүн маанилери да бүтүн константа болот.

2. Анык өзгөрмөлөрдүн ысымына башка символдор кошулбайт. Мисалы: `Z, LN, T3, R8`. Бул өзгөрмөнүн мааниси дайыма анык константа болот.

3. Символдук өзгөрмөнүн тиби `$` белгиси менен аяктайт.

Мисалы: `W$, AS$, U7$, KL$, CLASS10Z$`

Символдук өзгөрмөнүн мааниси катары клавиатурадан киргизилген же программа аткаруу процессинде калыптанган символ

дордун удаалаштыгы боло алат. Бир программада анык, бүтүн, символдук, өзгөрмөлөрдү бирдей ысым менен белгилөөгө болот. Ошондой эле өз тибин көрсөткөн символдор менен аталып бир эле программада орун алышы мүмкүн.

ИНДЕКСТҮҮ ӨЗГӨРҮЛМӨ ЖАНА АНЫН МААНИСИ

Бейсикте жөнөкөй өзгөрмөдөн тышкаркы индекстүү өзгөрмөлөр да колдонулат. Анда берилген сандардын тобу сандар таблицасы деп аталат. Таблицаны түзгөн ар бир сан таблицанын элементи болот. Таблицанын ысымы жөнөкөй өзгөрмөлөрдүн ысымындай түзүлөт, ал эми элементтери таблицанын ысымы боюнча аныкталат.

Мисалы: X=(21, 9, 190, 57, 85). Беш элементтен турган сандардын таблицасы сап түрүндө берилди. Бул таблицанын элементтери математикалык символикада белгилениши: 1-элементи X_1, 2-элементи X_2, 5-элементи X_5 деп аталат. Таблицанын элементинин ысымын көрсөткөн сан элементтин индекси деп аталат жана элементтин ордун көрсөтөт. Бейсикте таблицанын ысымынан кийин индекс "(" жана ") "символдору менен чектелип, аны менен бирдей деңгээлде жазылат. Мисалы жогорудагы таблицанын элементтери X(1), X(2), X(3), X(4), X(5) түрүндө жазылат. Мында X(1) таблицанын 1-элементи, ал 21 ге барабар, X(5) таблицанын 5-элементи, ал 85 ке барабар. Биз караган мисалды мамыча түрүндө берсек да болмок. Элементтери сап же мамыча түрүндө берилсе, анда ал бир өлчөмдүү, ал эми тик бурчтук түрүндө берилсе эки өлчөмдүү таблица деп аталат. Эки өлчөмдүү таблицанын элементтерин көрсөтүүдө биринчи саптын номери андан кийин мамычанын номери көрсөтүлөт. Мисалы A(1,3) – 1-саптагы 3-орундагы (мамычадагы) элементи, A(2,1) – 2-саптын 1-элементи.

Индекстер жалаң бүтүн константа менен берилбестен бүтүн, бүтүн өзгөрмө, бүтүн туюнтма түрүндө да берилет. Туюнтма түрүндө берилсе, анда анын маанисине жакын бүтүн санга тегеректелет.

Таблицанын элементтери символдук константа болушу мүмкүн. Символдук таблицалардын аталышы символдук өзгөрмөлөр сыяктуу эле "$" коолат. Мисалы: B$(10), KL$(4,4), Z1$(5,6)

Программа түзүүдө таблицанын элементи A(I,J) түрүндө каралат. A(I,J) менен таблицанын бардык маанилерин көрсөтүүгө болот. Ошондуктан мындай өзгөрмөнү индекстүү өзгөрмө деп айтабыз. Индекстерин минималдуу мааниси нөлгө барабар жана аларды өзгөртүү менен таблицанын бир элементинен экинчи элементине өтүүгө болот.

Өзгөрмолор эсте төмөндөгүдөй көлөмдү ээлейт:

№	Жазылышы	Максималдык	Минималдык
1	Өзгөрмөнүн ысымынын узундугу	40 символ	1 символ
2	Саптын узундугу	32762 символ	0 символ
3	Бүтүн	32767	-32768
4	Аныk(оң маанилер) Аныk(терс маанилер)	3.402823E+38 - 2.802597E-45	2.802597E-45 -3.402823E+38
5	Массивдин өлчөмү (бардык элементтери)	65535 байт	1 байт
6	Массивдин индекстик мааниси	32767	-32768
7	Процедуранын өлчөмү	64 Кбайт	0
8	Аргументтин саны	60	0
9	Берилиштер файлынын номери	255	1
10	Берилиштер файлынын жазылыш номери	2147483647	1
11	Берилиштер файлынын жазылыш өлчөмү	32 Кб	1 байт
12	Берилиштер файлынын олчөмү	дискте мүмкүн болгон мейкиндик	0

АЛГАЧКЫ ПРОГРАММА

Бейсик тилиндеги программа 65535 ке чейинки саптан турат. Бир сапка бир нече операторду ":" белгиси менен ажыратып жазууга болот. Операторлордун аткарылыш ирети солдон оңго карай аткарылат. Саптын максималдык узундугу 255 чейинки символдордон болушу мүмкүн (экрандан караганда 7 сапка жакын болот). Бир сапка бир нече оператор жазуу менен эсти үнөмдөөгө болот, ошондой эле программаны структуралык түргө келтирүүгө болот. Бирок мындай жазылышты окуу татаалыраак болуп калат. Ошондуктан ар бир операторду өзүнчө сапка жазуу ыңгайлуу. Программа менен иштөөдө окулушу ыңгайлуу болушу үчүн, программанын каалаган сабына түшүндүрмө жазуу ыңгайлуу. Программада каалагандай текстти, кызматчы сөздөрдү жазууга болот. Көлөмдүү программалардын бөлүктөрүн белгилоодо, колдонулуучу өзгөрмөлөргө түшүндүрмө берүүдө REM оператору колдонулат. Бул оператор аткарылбоочу оператор болуп эсептелет. оператордун жалпы форматы:

REM түшүндүрмө

REM операторунун ордуна " ' " белгисин колдонсо да болот

Мында, түшүндүрмодо каалаган Бейсиктин символу жазылат да түшүн

дүрмө катары кабыл алынат. Мисалы:

```
REM Алгачкы программа
PRINT "Мен компьютермин, кел таанышалы"
INPUT"Сенин атын ким? "; A$
PRINT A$, ", кел  достошолу !"
PRINT A$, " компьютер менен дос"
END
```

Ал эми программанын аякташы жогоруда мисалдан көрүнүп тургандай END оператору менен аяктайт. END-оператору ошондой эле кээ бир тармакталган операторлордун да аякташын көрсөтөт. Ошондой эле программаны кээ бир учурларда токтотуу талап кылынат. Бул учурда STOP оператору колдонулат. Мындай токтотуудан кийин программаны кайрадан улантууга болот. STOP операторунун END операторунан айырмасы STOPтон кийин көрсөтүлгөн номер боюнча программа ишин улантат. STOP то эскертүү берилбейт. Булар менен бөлүмдүн кийинки темаларында таанышa аласыңар.

Бейсик системасында F5 баскычын басуу менен бул программаны иштетебиз. Экранда "Мен компьютермин, кел таанышалы", "Сенин атың ким?" деген сүйлөмдөр пайда болот жана колдонуучудан анын ысмын киргизүүсүн компьютер күтөт. Клавиатуранын жардамы менен колдонуучунун ысмы киргизилет, мисалы: "Асида". Анда монитордун экранында "Асида, кел достошолу !" жана "Асида компьютер менен дос" деген сүйлөмдөр чыгарылат.

АРИФМЕТИКАЛЫК ТУЮНТМАЛАР

Кадимки математикада колдонулуучу туюнтмалар Бейсикте атайы жазылыштары менен берилип колдонулат. Туюнтмага катышкан чоңдуктар операнд деп аталат. Демек туюнтманы амал белгилери менен бириктирилген операнддар түзөт. Тактап айтканда константа, өзгөрмөлөр же константа-өзгөрмөлүү функциялардын амал менен байланышып, жыйынтыкта бир маани берүүчү комбинацияны ТУЮНТМА деп түшүнөбүз.

Туюнтманын операнддары – константалар жөнөкөй жана индекстүү өзгөрмөлөр, стандарттуу функциялар түрүндө берилет. Эгер туюнтмадагы операндалар арифметикалык амал символ белгилери менен берилсе, анда ал **арифметикалык туюнтма** деп аталат. Клавиатуранын информация киргизүү өзгөчөлүгү туюнтмаларды сап менен жазууну талап кылат. Буга ылайык арифметикалык амалдарды жазуу үчүн атайы символдор киргизилген:

+	кошуу амалы	^	даража белгиси
-	кемитүү амалы	\	бүтүн бөлүү белгиси

*	көбөйтүү	MOD	арифметикалык модуль
/	бөлүү		

Бул амалдардын жазылышына мисал келтирели:

Мисалдар	Бейсикте жазылышы	Мисалдар	Бейсикте жазылышы
-3	-X	4/0.2 = 20	X / Y
2^5=32	X ^ N	10\9 = 1, же 25.68\9=2	X \ Y
5.4*1.25=6.75	X * Y	13 MOD 3.8 = 1	X MOD Y
4+3.87 = 7.87	X + Y	5 – 3.78 = 1.22	X - Y

Программанын иштеши менен туюнтманы эсептөө, амалдардын аткарылыш ирети боюнча жүргүзүлөт. Адегенде кашаа, функция, даражага көтөрүү, көбөйтүү, бөлүү, MOD, кошуу, кемитүү. Ал эми бирдей иреттеги амалдар жазылыш ирети менен аткарылат.

Туюнтмалардын жазылышына мисал келтирели.

Мисалдар	Бейсикте жазылышы	Мисалдар	Бейсикте жазылышы
a	A	$1{,}64b$	1.64*B
$\frac{x^3}{y^5}$	X^3/Y^5	$\frac{a-b}{c-d}$	(A - B)/(C -D)
$\frac{k}{l \cdot m}$	K/(L*M)	$\frac{ab^2+c+d}{e}$	(A*B^2+C+D)/E
b^2-4ac	B^2-4*A*C	$\frac{x}{1+\frac{1}{y}}$	X/(1+(1/Y))

Туюнтманы эсептөөдө бир типтеги чоңдуктардан турган туюнтманын мааниси ошол эле типте болот. Мисалы: X%+25 туюнтмасынын мааниси бүтүн, ал эми A+3*B туюнтмасы анык мааниге ээ болот. Туура эмес жазылган туюнтмага мисал келтирели: 1)8X; 2)A*-B; 3)K*((3+SQR(X);

1-мисалда 8 менен X тин ортосунда арифметикалык амал болушу керек; 2-мисалда арифметикалык амал белгилерин катар жазууга болбойт; 3-мисалда ачылган кашаага ошончо жабылган кашаа туура келет.

СИМВОЛДУК ЖАНА ЛОГИКАЛЫК ТУЮНТМАЛАР

Операндалары символдук константалар, символдук өзгөрмөлөр, символдук функциялар болгон жана "+" же "&" амперсанд символдору менен байланышкан түрдүү комбинациялар символдук туюнтма деп аталат. Көрсөтүлгөн "+" жана "&" белгилери саптарды бириктирүү үчүн колдонулат. Мисалы: A$+B$ туюнтмасынын мааниси болуп A$ сабынын аягына B$ сабын бириктиргенде чыккан сап

эсептелет. Атап айтканда, " Айдар" + "бек " же "Айдар " & "бек " туюнтмасынын мааниси "Айдарбек" сөзү болот.

Эки арифметикалык же эки символдук туюнтманын ортосундагы катыштар логикалык туюнтманы түзөт. Логикалык туюнтма "чын" же "жалган" деген эки гана мааниге ээ боло алат. Мисалы X=3 болсо, анда X>2.8 туюнтмасынын мааниси "чын", себеби 3>2.8 барабарсыздыгы аткарылат. Эгер X=1 болсо, анда туюнтма "жалган" маанiге ээ болот. Салыштыруу жана катышты аныктоо үчүн катыш амалдары колдонулат. Катыш амалдарынын белгилери төмөндө көрсөтүлгөн:

белги	амал	мисал	белги	амал	мисал
=	барабар	A = B	< >	барабар эмес	A < > B
<	кичине	A < B	<=	кичине же барабар	A <= B
>	чоң	A > B	>=	чоң же барабар	A >= B

Катыш амалдары менен болгон логикалык амалдарга мисал:

Мисалдар	Бейсикте жазылышы	Мисалдар	Бейсикте жазылышы
A ≥ B	A>=B	0<X<3	0<X AND X<3
A = B	A=B, A$=B$	X>3 жана X ≠ 5	X>3 AND X<>5
A ≠ B	A<>B, A$<>B$	X> 8 же X<1	X>8 OR X<1
X ≠ 5	NOT (X=5)		

Бул мисалда колдонулган логикалык амалдарга түшүндүрмө бере кетели.

NOT – Аргументти логикалык тануу. Аргуметтин маанисин карама каршы белгиге өзгөртөт.

AND – Коньюкция жана аргументти логикалык көбөйтүү.

OR – Дизъюнкция, же эки аргументти логикалык кошуу.

Жогоруда мисалдарда көрсөтүлгөндөй салыштыруу бир типтеги мисалдарда гана жүргүзүлөт. Символдук жана арифметикалык мисалдарды салыштырууга болбойт. Символдук туюнтмаларды салыштырууда андагы символдор жазылуу тартиби боюнча солдон онду көздөй түгөйлөш салыштырылат. Мисалы, "БАР" жана "БАК" сөздөрүн салыштырганда "Б" жана "Б" тамгалары, андан кийин "А" жана "А" тамгалары, "Р" жана "К" тамгалары түгөйлөш салыштырылат. Салыштыруу биринчи дал келбей калган түгөйлөргө чейин жүргүзүлөт. "информатика" жана "информация" логикалык туюнтмаларынын биринчи жети символдору дал келет, салыштыруу "т" жана "ц" түгөйлөрүнө жеткенде бул логикалык туюнтманы салыштыруу токтотулат да, мааниси "жалган" болуп чыгат. Эгер салыштыруу аягына чейин дал келсе, анда логикалык туюнтманын мааниси "чын" болмок. Узундуктары түрдүү болгон символдук туюнтмаларды салыштырүү керек болсо, алардын кыскасын бош калтыруу " " символу менен толуктоо керек. Мисалы "Назарбек"="Назар " туюнтмасын салыштыруу алтынчы символго

чөпин жүргүзүлөт. Төртүнчү символ дал келбегендиктен бул логикалык туюнтма "жалган" маани алат.

Эскертүү: Шарты <, >, <=, >=, =, <> белгилери менен салыштырууда Бейсик алфавитинин тартиби эске алынат. Логикалык туюнтманын маанисин салыштырууга мисалдарды карайлы:
"ЭМИЛБЕК" > "ЭМИЛ ", "БАР" <>"ЖОК", "КАНАТ"="КАНАТ"

МАТЕМАТИКАЛЫК ФУНКЦИЯЛАР

Эсептөө маселелерин чыгарууда, функциялардын маанилерин таблица же логарифмдик сызгычтар колдонулуп эсептелинет. Ушул эле маселелерди компьютерде чыгарууда анда сакталган стандарт функциялардын библиотекасына кайрылабыз. Кеңири колдонулуучу функциялардын маанилерин эсептөөчү программалардын тобу стандарттык(программалык модуль) катары каралып, машиналык тилде транслятордо орнотулган. Стандарт функцияларды колдонууда анын аргументи менен функцияны берип жыйынтыгын көрө алабыз. Бейсикте стандарттык функциялар латындын үч тамгасы менен жазылып, аргумент тегерек кашаага алынат. Бейсикте колдонулуучу стандарттык функцияларды карайлы:

ABS(X) – X тин абсолюттук маанисин берет($|X|$).

ATN(X) – арктангенс х маанисин берет(arctg(x)). Бурчтун х ке барабар мааниси. Бурч -PI/2 ден PI/2 ге чейинки аралыкта радиандык бирдик менен эсептелет. Радиандан градуска которуу үчүн радиан= PI/180 колдонулат, PI/180=1.745329252E-2

EXP(X) – e^x ти берет, (e=2,718281828), X<=145.0628608586 (E^X) менен алмаштырууга болбойт, башкача айтканда даража катары кароого болбойт.

FIX(X) – X тин бөлчөк бөлүгүн алып таштайт.

INT(X) –X тен ашпаган чоң бүтүн N санын берет. Мында N<X<=N+1

LOG(X) – Хтин логарифми. X(lnX), мында X оң сан болушу керек. А негизи боюнча алгоритмди эсептөөдө а $\lg_a x$: $\lg_a x = \lg x / \ln a$ колдонуу керек.

RND(X) –0 дөн 1 ге чейинки ар кандай жайгашкан капилет сандарды чыгарат. X<0 функцияга ар бир кайра кайрылганда жаңы сандардын удаалаштыгын чыгарат. X=0 маанисинде мурунку капилет сандын маанисин көрсөтөт. X>0 бир эле сан удаалаштыгындагы капилет сан маанилерин чыгарат.

SGN(X) – X тин маанилерин томондогүдөй чыгарат:

-1, эгер X<0 0, эгер X=0 +1, эгер X>0

SIN(X) – X ти радиандык маани катары кабыл алып, X тин синусун (sin x) чыгарат

COS(X) – X ти радиандык маани катары кабыл алып, X тин косинусун (cos x) чыгарат

SQR(X) – X тин тамырын чыгарат. Мында X тин мааниси нөлдөн чоң болушу керек.

TAN(X) – X ти радиандык маани катары кабыл алып, X тин тангенсин (tg x) чыгарат

Эскертүү: Жогоруда көрсөтүлгөн функцияларды колдонууда колдонуучу тендештиктер.

$$sh(x)=\frac{e^{x}-e^{-x}}{2};\qquad ch(x)=\frac{e^{-x}-e^{x}}{2};\qquad th(x)=\frac{sh(x)}{ch(x)}$$

$$\arcsin(x)=arctg\frac{x}{\sqrt{1-x^{2}}};\qquad \arccos(x)=\frac{\pi}{2}-arctg\frac{x}{\sqrt{1-x^{2}}}$$

Бул таблицадагы тригонометриялык функциялар менен иштөөдө төмөндөгүлөрдү билип алуу керек. n ^{0} тагы бурчтун мисалы, косинусун табуу керек болсо, COS(PI*n/180) колдонуу керек.

Эскертүү: өндүк логарифмди эсептөөдө lg=1/ln*lnx колдонуу керек.

Бул функцияларды колдонууда келтирилүүчү мисалдарды карайлы:

ABS(-8.5)=8.5	INT(-3.7)=-4	LOG(2)=0.693147
FIX(-3.7)=-3	INT(3.7)=3	SQR(2)= 1.414
FIX(3.7)=3	INT(2.4568)=2	ATN(1)*4 = 3
SIN(0) = 0	COS(0) = 0	RND(1) = 0.456217789

Функцияга кайрылуу мисалдарын карайлы:

Математикада жазылышы:	Бейсикте жазылышы:
$4\cos\frac{y}{x}$	4*COS(Y/X)
$3\sin^{2}\frac{x}{5}$	3*(SIN(X/5))^ 2
$e^{a^{3}}$	EXP(A^3)
$\frac{ab+2c}{2c+\cos a}$	(A*B+2*C)/(2*C+COS(A))
$a-\sqrt{a+\sin^{2}x}$	A-SQR(A+(SIN(X)^2))
$\frac{-b+\sqrt{d}}{2a}$	(-B+SQR(D))/(2*)A
$\sqrt[3]{x-arctg^{2}x}$	(X-ATN(X)^2)^1/3

2. БЕЙСИК ТИЛИНИН ОПЕРАТОРЛОРУ

МЕНЧИКТӨӨ ОПЕРАТОРУ

Бейсиктин оператору компьютерде информация иштетүү аракеттерин атакаруу үчүн жазылышты, ал эми операторлордун удаалаштыгы программаны түзөт. Программа компьютерде маселелерди чечүүдө, алгоритмге карата белгилүү эрежелер менен түзүлөт, программанын иштеши менен жыйынтыгы компьютердин экранына чыгарылат. Керектүү учурда принтер аркылуу кагазга чыгарууга же эске сактап коюга болот. Компьютерде кандай гана маселе чыгарбайлы анын баштапкы маанилеин берүү менен иш башталат. Баштапкы маалыматтарды берүүдө LET менчиктөө оператору колдонулат. Жалпы жазылыш форматы: LET<**өзгөрмө**>=<туюнтма>

Мисалы: LET X=A Мында: X – озгөрмөнүн идентификатору;

A – арифметикалык же логикалык туюнтма.

Бул операторду иштеши менен А туюнтмасынын мааниси эсептелип жыйынтыгы X өзгөрмөсүнө туура келген эстин бөлүкчөсүнө жазылат б.а. X ке менчиктелет. А туюнтмасы арифметикалык, логикалык же символдук болушу мүмкүн. LET операторунун аткарылышына эң жөнөкөй мисал карайлы.

```
LET B%=10:LET C=5.38:LET D=2.05*B%+C
```

Бул программанын иштеши менен 10 саны В бүтүн өзгөрмөсүнө, 5.38 саны С анык өзгөрмөсүнө, үчүнчү саптагы туюнтманын мааниси эсептелип, D анык өзгөрмөсүнө менчиктелет.

Менчиктөө операторунун LET кызматчы сөзүн жазуунун зарылдыгы деле жок. Мисалы: K=81: M=K+1: Y=K+M-5

Бул операторлордун иштеши менен K өзгөрмөсүнүн алгачкы мааниси 81 ге, K га 1 саны кошулуп, M өзгөрмөсүнө берилет да анын мааниси 82 ге барабар болот. Акыркы операторду иштеши менен Y тин мааниси 158 ге барабар болот.

Менчиктөө опреаторундагы өзгөрмө оң жагындагы туюнтмага да катыша алат. I%=I%+1 операторунун иштеши менен I% өзгөрмөсү анын эстеги мааниси 8 болсо, анда ал 9 болуп калат. Көпчүлүк учурда бул жазылыш бир же бир нече операторлордун аткарылышын саноо үчүн колдонулат. Ошондуктан I% озгөрмөсүн санакчы деп аташат.

Менчиктөө операторунун оң жагындагы туюнмасынын мааниси ал менчиктелип жаткан сандык өзгөрмөдөн башка типте болушу мүмкүн. Мындай учурда оң жагындагы тиби сол жагындагы тибин озгөрот. Мисалы: P%=6.2 болсо P% тин мааниси тегеректелип 6 болот

P=48 операторунун иштеши менен P өзгөрмөсүнө 48.0 аныk туraктуу маани менчиктелет. G=F*S-2*K туюнтмасынын мааниси аныk болот. Эгер G%=F*S-2*K болсо, бүтүн N=T+M%+N% туюнтмасынын маанисин аныkка өзгөртөт.

Символдук менчиктөө операторунун сол жагында символдук өзгөрмө, оң жагында символдук туюнтма болот.

Мисалы: LET F$="QBasic", F$ символдук өзгөрмөсүнө QBasic символдук мааниси туура келет.

```
LET X$="Улуттук":LET Y$="компьютердик"
LET Z$="гимназия":LET K$=X$+Y$+Z$
```

Бул саптардын удаалаштыгынын аткарылышы менен саптык өзгөрмөлөргө тиешелүү түрдө "Улуттук", "компьютердик", "гимназия" сөздөрү менчиктелет да, акыркы саптын иштеши менен алардын биригүүсү келип чыгат.

МААЛЫМАТТАРДЫ БЕРҮҮ ОПЕРАТОРЛОРУ

Бейсик тилинде маалыматтарды компьютердин эсине берүүнүн эки жолу кездешет. Алардын бири READ, DATA операторлорунун жардамы менен жүргүзүлөт. Бул операторлор программада түгөйү менен колдонулат жана алардын жардамы менен өзгөрмө-лөрдүн маанилери программанын текстинде көрсетүлөт. Жазылыш форматы: DATA <тураkттуулардын тизмеги>

READ <өзгөрмөлөрдүн тизмеги>

Мында: <тураkттуулардын тизмеги> жана <өзгөрмөлөрдүн тизмеги> "," белгиси менен ажыратылып жазылат жана текстик тураkтуулар тырмакчага алынбайт. Программадагы READ операторуиштегенге чейин, DATA операторундагы маалымат кабыл алынбайт. Ошондуктан DATA операторун программанын каалаган сабына жайгаштырсак болот. READ оператору аткрылган кезде бул операторду көрсөтүлгөн ысымдар боюнча DATA операторунда берилген тураkтуу маанилерди ирети менен эс уячасына бөлүштүрүп жазат. DATA жана READ операторлорундагы берилиштер бирдей снда болушу керек, DATA операторундагы тураkтуулардын саны READ операторундагы өзгөрмөлөрдүн санынан аз болбошу керек. Эгер DATA операторундагы маалыматтардын саны READ операторундагы өзгөрмөлөрдүн санынан ашык болсо, анда кийинки READ операторундагы өзгөрмөлөргө менчиктелет.

1-Мисал:

```
DATA 1,5,7,24,32,16,81,10
READ A, B                    {A=1, B=5}
.............................
READ C, D, E                 {C=7, D=24, E=32}
.............................
READX, Y, Z                  {X=16, Y=81, Z=10}
```

Кээ бир учурда DATA операторундагы берилиштерди кайтадан колдонуу зарылдыгы болот, анда RESTORE операторун колдонууга туура келет. RESTORE оператору DATA дагы берилиштерди калыбына келтирет. Жогорудагы мисалда RESTORE операторун колдонуп карайлы. 2-Мисал:

```
DATA 1,5,7,24,32,16,81,10
READ A, B                          {A=1, B=5}
.............................
READ C, D, E                       {C=7, D=24, E=32}
.............................
READ X, Y, Z                       {X=16, Y=81, Z=10}
RESTORE
READ K,L,M,N                       {K=1, L=5, M=7, N=24}
```

3-Мисал:

```
READ C$,S$,Z
DATA СНГ,БИШКЕК,720082
```

Кээ бир учурда калбына келтирүү баштапкы элементке келбестен, DATAнын көрсөтүлгөн номери боюнча башталышы мүмкүн. Мисалы (көрсөткүчтү которуу):

RESTORE 5 – көрсөткүчтү блоктун 5-маанисине келтирүү

RESTORE +6 – көрсөткүчтү учурдагы абалдан 6 позицияга жылдыруу.

RESTORE –3 – үчүнчү позицияга артка жылдыруу деп түшүндүрүлөт.

Маалыматтар блогун сандык, текстик көрсөткүчтү бөлүп көрсөтүү кабыл алынган.

1.RESTORE # – сан маалыматтар блогун баштапкы абалга келтирүү

2.RESTORE $ – текстик маалыматтар блогун баштапкы абалга келтирүү

3.RESTORE – эки көрсөткүчтөгү маалыматтар блогун баштапкы абалга келтирүү.

Маалыматтарды берүүнүн дагы бир жолу INPUT операторунун жардамы менен ишке ашырылат. INPUT– мааламаттарды клавиатурдан берүү оператору. Программанын иштөө учурунда эсептөөлөрдө колдонулуучу сан(арифметикалык) же текстик озгөрмөлөрдүн маанилерин клавиатурадан киргизүүнү уюштурат.

```
INPUT p1, p2, ... , pn
```

Бейсик системасы INPUT операторун аткруу менен экранга "?" белгисин чыгарат. Жооп берүү жеңил болуш үчүн INPUT операторунан мурун же INPUT операторун өзүндө көрсөтүлүүчү маалыматтын саны жана тиби боюнча түшүндүрмө берүү сунуш кылынат. Киргизилүүчү маалыматтар жана алардын тиби INPUT операторундагы өзгөрмолордун тизмеги менен дал келиши керек. Эгер мындай туура келүүчүлүк болбой калса, анда система ката кетирилген сандан баштап кайрадан

киргизүүнү талап кылат. Бул операторду иштетишине жөнөкөй мисалы карайлы.

INPUT A, B%, C$ оператору A өзгөрмөсү үчүн анык, B% үчүн бүтүн, C$ үчүн символдук маанилерди киргизүүнү талап кылат. Программа иштөөдө, INPUT жолугаары менен "?" белгиси чыгып, A, B%, C$ өзгөрмөлөрүнүн маанилерин күтөт. Клавиатурадан тиешелүү маанилерин "," менен ажыратып киргизебиз. Анда жогоруда айтылгандар экранда төмөнкү формада орун алат. Мисалы:

```
? 8.9,2,китеп
```

Бул маанилерди бергенде өзгөрмөлөр A=8.9,B%=2,C$="китеп" маанилерин алышат. 2-Мисал:

```
INPUT "Мектеп";F$, "кошун";B$,"Класс";C
```

Бул саптардын иштеши менен тиешелүү түрдө "УКГ", "Манас", "10" маанилерин киргизсек болот.

Ошондой эле LINE INPUT операторун да колдонсок болот. LINE INPUT операторунун INPUT операторунан айырмасы клавиатурадан же файлдан 255 чейинки символдогу текстти же программаны киргизүү мүмкүнчүлүгүндө болуп саналат. Символдорду киргизүү ENTER баскычын басуу менен аяктайт. Бул операторлордун төмөндөгүдөй жазылыш форматтары колдонулат:

INPUT "<чакыруу>", <**өзгөрмөлөрдүн** тизмеги>

Мында: <чакыруу> – маалымат киргизүүдө экранда көрсөтүлүүчү сап; <**өзгөрмөлөрдүн** тизмеги> – клавиатурадан киргизүүчү маалымат сактоочу бир нече өзгөрмө(үтүр менен ажыратылган), тамга менен башталып, 40 символдон турушу мүмкүн;

МААЛЫМАТТАРДЫ ЧЫГАРУУ ОПЕРАТОРУ

Маселелерди чыгарууда анын жыйынтыгын же түшүндүрмө текстерди экранга, принтерге чыгарууда же файлга жазууда PRINT оператору колдонулат. Жазылыш форматы:

PRINT <чыгарылуучу өзгөрмөлөрдүн тизмеси>

Мында: <чыгарылуучу өзгөрмөлөрдүн тизмеси> – сандык же символдук туюнтмалар же алардын жыйындысы.

Мисалы: PRINT F$,P,W%,7*A+C^2

Бул операторду иштетишинде F тин символдук, P нын анык, W пун бүтүн маанилери жана туюнтманын жыйынтыгы чыгарылат. Өзгөрмөлөр "," белгиси менен ажыратылып жазылгандыктан, 5 зонага бөлүнгөн 14 позиция аралыгында чыгарылат. Маалыматтарды мындай чыгаруу позициялык форматта чыгаруу деп аталат. Мисалы:

```
A=5:B=10:P=21
     PRINT A,B
```

```
PRINT P
```

Бул программанын иштеши менен

```
5    10
21
```

түрүндөгү чыгарылышты көрөбүз. Ал эми ";" менен ажырытылыш жазылса, анда тыкыс форматта чыгарылат. Эгер PRINT операторунан кийин эч нерсе жазылбаса, анда бир сап бош калтырылат.

```
PRINT
PRINT
PRINT
```

саптары иштеши менен үч сап бош калтырылат. TAB (аргумент, туюнтма) кызматчы сөзүн PRINT менен бирге колдонуп жазылыштарды ыңгайлуу чыгарууга болот. Мында туюнтма аргумент көрсөтүлгөн позициядан баштап чгарылат.

Мисалы:
```
PRINT TAB(20);"5"
PRINT TAB(5);"5";TAB(12);"98"
PRINT TAB(20);"*";TAB(30);"0"
```

Бул программалык саптардын биринчисинин иштеши менен 20-позицияга 5 саны чыгат, экинчисинин иштеши менен 5-позицияга 5 саны, 12-позицияга 98 саны чыгат, үчүнчүсүнүн иштеши менен 20-позицияга "*" белгиси, 30-позицияга 0 саны чыгат.

Маалыматтарды берүү жана чыгаруу операторлорун пайдаланып каалагандай сандын керектүү процентин табуу мисалын карайлы:

```
input "сан киргиз",   n
input "канча проценти керек"; p
x=n/100*p
print n "санынын"; p; " проценти x=";x
```

LPRINT маалыматтарды <LPT1> принтерге чыгарат. Жалпы жазылыш формасы: LPRINT <туюнтма>[{:,;}]

<туюнтма> –печатка чыгарылуучу бир же бир нече сан же символдук туюнтма. [{:,;}]–кийинки чыгарылыш кайдан башталарын көрсөтөт

; – чыгаруу акыркы мааниден кийин чыга баштайт

, – чыгаруу кийинки зонанын башталышында чыгарылат.

Мисалы:
```
REM бул мисал иштеши үчүн принтер керек
LPRINT:FOR I%=1 TO 20
LPRINT I%;
IF LPOS (1)>=10 THEN LPRINT 'жаңы сап баштоо
NEXT I%
```

PRINT маалыматтарды экранга чыгарат же файлга жазат. LPRINT LPT1 принтеринде маалыматтарды чыгарат.

LPOS(n%) – чыгарууга жиберилген символдордун санын калыбына келтирет. Мында, n% – принтер портторунун бирин көрсөтөт

0- LPT1, 1=LPT1. 2=LPT2, 3=LPT3

PRINT USING форматталган маалыматтарды экранга же файлдарга жазат: PRINT USING "<формат>",<туюнтма>[{; :,}|
<формат> - символдор; формат аныкталуучу бир же андан көп символдук туюнтма. <туюнтма> – үтүр, үтүрлүү чекит, табуляция же пробел менен ажыратылган бир же бир нече сан, же символдук туюнтма. {; :,} – кийинки чыгарылыштардын башталышын көрсөтөт
; – чыгаруу акыркы мааниден кийин чыга баштайт
, – чыгаруу кийинки зонанын башталышында чыгарылат.

Мисал:
```
A=152.3765
PRINT USING "###. ##"; A
LPRINT USING "+###. ####"; A
A$="KGJFHDN":PRINT USING "!"; A$
LPRINT USING "\\"; A$
```

Экинчи сап иштеши менен A нын мааниси экранда 152.37; үчүнчү сап иштеши менен принтерге 152.3765 саны; бешинчи саптан K; 6-саптан принтердеги кагазга KJ чыгарылат.

GOTO ШАРТСЫЗ ӨТҮҮ ОПЕРАТОРУ

Компьютер GOTO операторун аткаруу менен көрсөтүлгөн саптагы операторго башкарууну өткөрүп берет:
GOTO <оператор же операторлордун тизмеги>

Башкарууну шарт менен өтүү татаал структурада бир же бир нече операторлорго берилет. Бир нече операторлордун жазылышы ":" белгиси менен улантылат. Алардын саны саптагы символдордун саны менен чектелет. Бизге мурдатан белгилүү болгондой саптагы символдордун саны 255 тен ашпашы керек.

Мисалы: 1) A=5:GOTO 50:PRINT A

Мында PRINT оператору аткарылбайт, башкаруу же программанын аткарылышы 50-сапка өтүү менен, 50-саптагы операторлор аткарылып, программанын иштеши улантылат.

ON GOTO ӨТҮҮ ОПЕРАТОРУ

Жазылыш форматы:
ON <арифметикалык туюнтма> GOTO оператор1, оператор2, ...

ON GOTO операторунун иштеши менен башкаруу туюнтманын маанисине жараша өтөт. Туюнтманын мааниси бирге барабар болсо, анда программа тизмектин биринчи сабына барат, туюнтма 2 ге барабар болсо тизмектин 2-сабына барат. Эгер туюнтма көрсөтүлгөн

номерден башка маани алса, анда башкаруу ON GOTOдон кийинки сапка өтөт. ON GOTOдон кийин жок дегенде бир сап болушу керек.

Мисалы:

1)
```
INPUT X
ON X GOTO 200, 300, 400
PRINT
200 PRINT X+500
300  PRINT X+600
400 PRINT X+700
```

2)
```
PRINT " Аянт  табуу": PRINT "Төмөндөгүдөн танда"
PRINT "1.квадрат, 2.тегерек, 3. Үч бурчтук"
INPUT P,"маанисин киргиз:";A
ON P GOTO 10,20,30
PRINT
10 PRINT"квадраттын аянты"; A*A
20 PRINT"тегеректин аянты"; 3.14*A*A
30 PRINT" Үч бурчтуктун аянты"; A*A*SIN(3)/2
END
```

3)
```
REM Сүйлөшүү
PRINT "Саламатсызбы окуун кандай?"
INPUT A$:IF A$="Эн жакшы" THEN GOTO 10
PRINT "Аракеттен"
GOTO 20
10 PRINT "Сени дайыма ийгилик күтөт! "
20 END
```

4) Беш кабаттуу үйдүн ар бир кабатында 4 төн квартира бар. Ар кабаттагы квартиранын номерин өзүнчө чыгаруунун программасын түзүү керек.

```
PRINT "Кабаттын (этаждын) номерин киргизиңиз"
INPUT N:PRINT N
PRINT "Кабаттагы квартиралардын номерлери", N
ON N GOTO 20,40,30,50,10:STOP
10 PRINT "17,18,19,20":STOP
20 PRINT "1,2,3,4":STOP
30 PRINT "9,10,11,12":STOP
50 PRINT "13,14,15,16"
END
```

Программанын жыйынтыгы.

Эгер, N=1 болсо, 2-кабаттагы квартиралардын номерлери 5,6,7; N=2 болгондо, 3-кабаттагы квартиралардын номерлери 9,10,11,12; N=3 болгондо, 4-кабаттагы квартиралардын номерлери 13,14,15,16 чыгарылат.

ШАРТТУУ ОПЕРАТОРЛОР

Биз буга чейинки таанышкан программалар саптардын удаалаштыгы менен биринен кийин экинчиси аткарылып келген. Программанын мындай түзүлүшү сызыктуу структурадагы программалоо деп аталат. Ал эми маселелерди аткарууда дайыма эле бул жол колдонула бербейт. Алгоритм бөлүгүндө таанышкандай маселенин берилишинде кандайдыр бир шарт коюлса, мындай маселелерди чыгарууда шарттуу операторлор колдонулат. Шарттуу операторлор менен иштөөнү программалоодо тармактуу структурадагы программалоо деп айтылат.

Шарттуу өтүү операторлорун толук шартуу жана толук эмес шарттуу өтүү катарында бөлүп карайбыз:

1. Толук шарттуу өтүү:

```
IF <шарт> THEN <оператор1> ELSE <оператор 2>
```

Эгер анда антпесе

Компьютер бул операторлорду аткаруу менен IF сөзүнөн кийинки шарттын аткарылышын текшерет. Эгер шарт аткарылса, анда башкаруу THEN кызматчы сөзүнөн кийин жайгашкан операторго берилет, шарт аткарылбаса, анда башкаруу ELSE кызматчы сөзүнөн кийин жайгашкан операторго берилет.

2. Толук эмес шарттуу өтүү:

```
IF <шарт> THEN <оператор1>
```

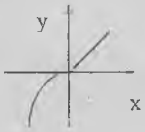
Бул учурда шарт аткарылса, башкаруу THEN кызматчы сөзүнөн кийин жайгашкан операторго берилет. Эгер шарт орун албаса, анда бул операторлордон кийинки турган сап аткарылат.

```
IF <шарт> GOTO N
```

Бул учурда шарт аткарылса, башкаруу N–сапка берилет, ал эми шарт орун албаса, анда бул операторлордон кийинки турган сап аткарылат.

1-Мисал: Аргументтин мааниси x тен көз каранды болгон $f(x)$ функциясынын маанисин тап.

$$f(x)=\begin{cases} x, & \text{эгерде } x>0 \\ 0, & \text{эгерде } x=0 \\ -x^2, & \text{эгерде } x<0 \end{cases}$$



Бул функциянын чыгарылышын төмөнкүчө жазууга болот:

```
IF X>0 THEN Y=X
IF X=0 THEN Y=0
IF X<0 THEN Y=-X^2
```

2-Мисал: Бүтүн x, y, z сандары берилген. 1) MAX(x+y+z, xyz) ти эсептегиле; 2) MIN(x,y,x·y) ти эсептегиле;

```
1) REM
```

```
   INPUT X%,Y%,Z%
   IF X+Y+Z>X*Y*Z THEN MAX=X+Y+Z ELSE MAX=X*Y*Z
   PRINT "MAX=";MAX
   END
2) REM
   INPUT X%,Y%,
   IF X<Y THEN MIN=X ELSE MIN=Y
   IF X-Y<MIN THEN MIN=X-Y
   PRINT "MIN=";MIN
   END
```

3-Мисал: a, b, c анык сандары берилген. a<b<c барабарсыздыгы аткарыла тургандыгын аныкта.

```
REM
INPUT A,B,C
IF A<B AND B<C THEN PRINT "ТУУРА" ELSE PRINT "ТУУРА
ЭМЕС"
END
```

Акыркы мисалда биз AND операторун колдондук. Бул оператор текшерилип жаткан шарттын чын же жалган экендигин айкындайт, эгер шарт "чын" мааниге ээ болсо, анда THEN ден кийинки оператор аткарылат, антпесе ELSE ден кийинки оператор иштейт.

Бейсикте тармактуу алгоритм менен иштөөдө тармак ичинде тармак уюштурууга болот. Үч A, B, C сандарынын чоңун табуу мисалын карап түшүнолу.

```
REM
INPUT A,B,C
IF A>B AND A>C THEN MAX=A ELSE IF B>C THEN MAX=B ELSE
MAX=C
PRINT "MAX=";MAX
END
```

ЦИКЛ ОПЕРАТОРЛОРУ, АЗЫРЫНЧА ЦИКЛИ

"Цикл" деген сөздүн өзүнө төмөнкүдөй түшүнүк берүүгү болот. Эгер кандайдыр кубулуш бир багыттын ичинде мейкиндиктин бир жеринен баштап кайрадан ошол жеринде аяктаса, андай кубулуш бир цикл жасайт. Компьютерде маселени чыгарууда IF THEN ELSE, GOTO операторлору менен цикл уюштурууга болот.

Бул учурда шарт текшерүү, программанын башында же аягында аткарылышы менен, **цикл азырынча** жана **цикл чейин** болуп экиге бөлүнөт. Операторлордун кайталанышы шартка чейин аткарылса, анда **цикл чейин**, эгер операторлордун кайталанышы шарттан кийин

аткарылса, анда **цикл азырынча** деп түшүнөбүз. Бул циклдардын жазылыш конструкциясына токтололу.

Азырынча циклинин конструкциясы:

```
...
n IF <шарт> THEN n+1
<оператор1>
<оператор2>
...
<операторN>
GOTO n
n+1 <циклдан чыгуу>
```

Азырынча циклинин иштешин мындайча карасак болот: "Азырынча шарт аткарылбаса, анда цикл операторлору аткарылсын". Качан гана шарт аткарылса, цикл өзүнүн иштешин токтотот да, акыркы жыйынтыкты берет. Эгерде циклдын биринчи кайрылуусунда шарт орун алса, анда цикл бир да жолу аткарылбайт. Кененирээк түшүнүк алуу үчүн төмөнкү мисалдарга кайрылалы(бул мисалдар менен Паскаль тилиндеги цикл операторлорун карап жатканда таанышканбыз):

1-Мисал: Эгерде суммалоо 3 төн баштапса жана ар бир кошулуучу мурдагыга караганда 2 ге чоң болсо, суммасы 200 дөн ашпаган натуралдык сандардын санын аныктагыла.

```
REM'ЦИКЛ АЗЫРЫНЧА
S=0:K=3:T=0
10  IF S>=200 THEN 20
S=S+K:K=K+2:T=T+1:GOTO 10
20 PRINT "кошулуулардын саны=";T
END
```

2-Мисал. Биринчи 50 натуралдык сандын суммасын эсептеп чыгаруу.

```
REM'ЦИКЛ АЗЫРЫНЧА
I=1:S=0
10 IF I>=50 THEN 20
S=S+I:I=I+1:GOTO 10
PRINT "Сумма =";S
END
```

3-Мисал. 20 дан 70 ке чейинки жуп сандардын суммасын эсепте.

```
REM'ЦИКЛ АЗЫРЫНЧА
K=20:S=0
10 IF K>=70 THEN 20
S=S+K:K=K+2:GOTO 10
PRINT "сумма=";S
END
```

Азырынча циклин жогоруда келтирилген операторлордон сырткары WHILE-WEND операторлорунун жардамы менен да берсек болот. Бул оператор берилген шарт орун алган убакта, ал шартты

канааттандырган операторлордун удаалаштыгын иштетет. Ал төмөнкү формада берилет:

```
WHILE <шарт>
...
WEND
```

Мисалы: суммасы 100 дөн ашпаган так сандардын суммасын тапкыла. Азырынча цикл канча жолу аткарылаары белгисиз. Программанын көрүнүшү:

```
S=0:I=0
WHILE S<=100:I=I+1:S=S+(2*I-1):WEND
PRINT I, S
```

Эгерде `WHILE S>=0` болсо, анда цикл чексиз болмок, анткени `S>=0` шарты дайыма орун алат.

Берилген шарт аткарылган убакка чейин цикл операторлорун бир нече жолу кайталоону `DO-LOOP` операторлорунун жардамы менен да берсек болот. Ал оператордун жазылыш форматы:

`DO WHILE <ШАРТ>`	`DO`
`<операторлордун тобу>`	`<операторлордун тобу>`
`LOOP`	`LOOP WHILE <ШАРТ>`

Мисалы: Эгерде суммалоо 3 төн башталса жана ар бир кошулуучу мурдагыга караганда 2 ге чоң болсо, суммасы 200 дөн ашкан натуралдык сандардын санын аныктагыла.

```
REM ЦИКЛ АЗЫРЫНЧА
S=0:K=3:T=0
DO WHILE S<=200:S=S+K:K=K+2:T=T+1:LOOP
PRINT "T=";T
END
```

ЧЕЙИН ЦИКЛИ

Кандайдыр бир шарт аткарылганга чейин кээ бир эсептөөлөрдү бир нече жолу аткарууга туура келсе, чейин циклинин конструкциясын колдонсок болот. бул циклдин өзгөчөлүгү, ал дайыма жок дегенде бир жолу аткарылат. Чейин циклинин конструкциясы төмөнкүдөй болот:

```
...
n] <оператор1>
IF <шарт> THEN n
<циклдан чыгуу>
```

Бизге түшүнүктүү болсун үчүн мурунку бөлүмдө(Турбо-Паскаль) каралып кеткен мисалдардын Бейсикте чыгарылышына токтололу.

1-Мисал: Кийинки сандын квадраты 625 тен чоң болгон убакка чейинки натуралдык сандардын квадраттарынын суммасын эсептегиле

```
REM ЦИКЛ ЧЕЙИН
```

```
S=0:K=1
10 S=S+K^2:K=K+1
IF K^2<625 THEN 10
PRINT "S="; S
END
```

2-Мисал. Биринчи 50 натуралдык сандын суммасын эсептеп чыгаруу.

```
REM'ЦИКЛ ЧЕЙИН
I=1:S=0
10 S=S+I:I=I+1
IF I<=50 THEN 10
PRINT "Сумма =";S
END
```

3-Мисал. 20 дан 70 ке чейинки жуп сандардын суммасын эсепте.

```
REM'ЦИКЛ ЧЕЙИН
K=20:S=0
10 S=S+K:K=K+2
IF K<=70 THEN 10
PRINT "сумма=";S
END
```

Ошондой эле чейин циклин DO-LOOP операторлорунун жардамы менен да берсек болот. Бул операторлор шарт аткарылбай калган учурга чейин цикл операторлорунун иштешин камсыз кылат:

```
DO UNTIL <ШАРТ>
<операторлордун тобу>
LOOP
```

```
DO
<операторлордун тобу>
LOOP UNTIL <ШАРТ>
```

Мисалы: Кийинки сандын квадраты 625 тен чоң болгон убакка чейинки натуралдык сандардын квадраттарынын суммасын эсептеп чыгалы.

```
REM'ЦИКЛ ЧЕЙИН
S=0:K=1
DO:S=S+K^2:K=K+1:LOOP UNTIL K^2>625
PRINT "S=";S
END
```

ПАРАМЕТРЛҮҮ ЦИКЛДЫ УЮШТУРУУ

Цикл уюштурууда аракеттердин кайталануу саны белгилүү болсо, FOR TO NEXT операторлору колдонулат. Бул операторлордун жазылыш форматы:

```
FOR        J=A     TO        B       STEP C
```

үчүн баштапкы маани чейин акыркы маани өсүндү

```
<ОПЕРАТОРЛОРДУН ТОБУ>
NEXT J{циклдын аяты}
```

Бул операторлордун иштеши менен J – цикл параметри (өзгөрмөсү) баштапкы маанини алат да, циклдин тууралыгы текшерилет. Циклдын параметри ошол аралыкта жайгашуу шарты аткарылса, FOR менен NEXTтин ортосундагы операторлор аткарылат. Кадам(өсүндү) чоңдугу кошулгандан кийин параметр жаңы маанини алып, операторлордун блогу көрсөтүлгөн сан боюнча кайталанат.

Циклдын аткарылышында төмөнкүлөрдү эстен чыгарбоо керек:

1. Эсептегич I ге баштапкы гана маани менчиктелерин унутпоо керек(биздин учурда ал A га барабар)
2. Эгерде баштапкы маани акыркы маани менен дал келсе, цикл операторлору бир гана жолу аткарылат.
3. Эгерде баштапкы маани акыркы мааниден чоң болсо жана өсүндүнүн мааниси оң сан болсо, анда цикл оператору бир да жолу аткарылбайт.
4. Эгерде баштапкы маани акыркы мааниден чоң болсо жана өсүндүнүн мааниси терс сан болсо, анда цикл операторлору аткарыла берет.
5. Эгерде өсүндүнүн мааниси бирге барабар болсо(C=1), анда STEP операторун жазбай койсок да болот.
6. Циклден чыгууда эсептегичтин(счетчиктин) мааниси акыры маани менен дал келет: I=B

1-Мисал: 10 дон 20 га чейинки натуралдык сандардын квадратын экранга чыгар.

```
FOR I=10 TO 20:A=I^2:PRINT I;"санынын квадраты=";A);
NEXT I
```

2-Мисал: [2; 10] аралыгында $y=\frac{5x^3+\sqrt{4x+10}}{|x^5+x-7|}$ функциясынын маанисин эсептегиле. x тин мааниси улам 0,5 ке чоңоёт деп карагыла.

```
FOR X=2 TO 10 STEP 0.5
Y=(5*X^3+SQR(4*X+10))/ABS(X^5+X-7):PRINT "X=";X,"Y=";Y
NEXT X
```

Цикл менен иштөөдө бир циклдын ичине башка цикл, б.а. кийиштирилген цикл уюштурууга болот. Анда негизги цикл сырткы цикл, анын ичине уюштурулган цикл ички цикл деп аталат. Циклдарды мындай уюштурууда сырткы жана ички циклдин параметрлери айырмаланышат. Сырткы циклдин параметринин бир мааниси үчүн ички циклдин параметринин бардык маанилери аткарылууга тийиш.

```
FOR  I=A  TO B
...
FOR  J=A1  TO B1
...
FOR  K=A2 TO B2
<операторлор тобу>
NEXT K,J,I
```

БАЗАЛЫК АЛГОРИТМДЕР

Бул темада биз мындан мурунку бөлүмдө (Турбо-Паскаль) карап кеткен базалык алгоритмдердин Бейсик тилинде колдонулуштарын карап чыгабыз. Ал алгоритмдердин аткарылуу жолдору мурунку бөлүмдө кенен баяндалган, аларга токтолуп отурбай эле Бейсик тилиндеги прораммасын түзүү менен гана чектелебиз.

n сандын суммасын жана көбөйтүндүсүн эсептеп, чыгаруучу алгоритм.

```
REM'summa
S=0;P=1:FOR K=1 TO 10 :S=S+K:P=P+I:NEXT K
PRINT "summa=";S,"koboytundu=";P
END
```

Берилген шартты канааттандыруучу сандардын эсебин аныктоо.

1-Мисал: клавиатурадан киргизилген 30 сандын арасынан терс сандардын санын аныктагыла.

```
REM'ШАРТ
INPUT N:K=0
FOR I=1 TO N:PRINT "A нын маанисин бер":INPUT A
IF A<0 THEN K=K+1:NEXT I
IF K=0 THEN PRINT "терс сандар жок" :GOTO 10
PRINT "терс сандардын саны="; K
10 END
```

2-Мисал: Берилген сандардын удаалаштыгынан оң жана терс элементтердин суммасын эсептегиле.

```
REM'ШАРТ
INPUT N:K=0:T=0
FOR I=1 TO N:PRINT "A нын маанисин бер":INPUT A
IF A>0 THEN K=K+1 ELSE T=T+1:NEXT I
IF K=0 THEN PRINT "оң сандар жок"
IF T=0 THEN PRINT "терс сандар жок" :GOTO 10
PRINT "Сумма=";K+T
10 END
```

n! ди эсептөөнүн алгоритми

```
REM'ФАКТОРИАЛ
INPUT N:F=1;
FOR I=1 to N do:F=F*I:NEXT I
PRINT N;"!=";F
END
```

БИР ӨЛЧӨМДҮҮ МАССИВДЕР

Массив жөнүндө китептин II бөлүгүндө кенен түшүнүк берилген. Биз аларга токтолуп отурбай эле бир өлчөмдүү массивдерди Бейсик тилинде баяндоо, иштетүү аракеттери менен таанышып чыгалы.

Программалоодо массивдин элементтерин колдонгонго чейин аны баяндоо керек, ал "Dimension" - өлчөмдүүлүк сөзүнөн алынган DIM оператору менен баяндалат. Бир сапта бир нече массивди баяндоого да болот. Массивдин өлчөмүн баяндоодо арифметикалык туюнтмаларды колдонсок да болот, анда бөлчөк бөлүгү каралбайт. Жазылуу формасы: DIM A1(N),A2(K) Мында: A1,A2 – массивдин ысымдары, N,K – индекстеринин чектери.

Мисал: DIM A(20)- 20 элементтен турган А массиви.

Эмесе бир өлчөмдүү массивди иштетүүгө мүмкүн болгон базалык алгоритмдерди санап өтөлү:

1. Бир олчомдуу массивдин маанилерин берүү
2. Бир олчомдуу массивдин маанилерин экранга чыгаруу
3. Бир олчомдуу массивдин элементтеринин суммасын эсептөө
4. Берилген шартты канааттандырган бир өлчөмдүү массивдин элементтеринин санын эсептөө
5. Бир олчомдуу массивдин максималдык(минималдык) элементин жана аның катар номерин аныктоо

Жогоруда көрсөтүлгөн базалык алгоритмдердин ар бирине өзүнчө токтололу.

1. Бир олчомдуу массивдин маанилерин берүүнү
   - маанини клавиатурадан киргизүү менен;
   - формула боюнча жүргүзсөк болот.

Эки учурда тең биз цикл уюштуруу менен иш алып барабыз.

Мейли биз 5 элементтен турган массивге маани берели.

- бир олчомдуу массивге клавиатурадан маани берели.

```
FOR I=1 TO 5 :PRINT "элементти киргизгиле"
INPUT B(I) 'массивдин элементтерин киргизүү
NEXT I
```

Ошондой эле массивдин элементтерин киргизүүнү READ-DATA операторлору менен да ишке ашырсак болот:

```
FOR I=1 TO 5:READ B(I) 'массивдин элементтерин окуу
NEXT I
DATA 3,6,-6,0,19
```

Бир эле учурда бир нече массивди удаалаш клавиатурадан киргизүүгө болот. Мында ар бир массивге өзүнчө цикл уюштурууга болот же баарысы үчүн бир эле цикл уюштуруу жетиштүү болуп саналат. Бул учурда массивдин элементтерин киргизүү ирети менен жүргүзүлөт. Б.а

1-массивдин 1-элементи, андан кийин 2-массивдин 1-элементи,ж.б. N-массивдин 1-элементи киргизилүүгө тийиш. Кийинки элементтерди киргизүү ушул тартипте оруп алат:

```
FOR I=1 TO 20:PRINT "элементти киргизгиле"
INPUT B(I),C(I) 'массивдин элементтерин киргизүү
NEXT I
```

Эгерде массивдин элементтеринин саны бири-бири менен дал келбесе, анда ар бир массив үчүн өзүнчө цикл уюштурулушу зарыл.

```
FOR I=1 TO 20:INPUT B(I):NEXT I
FOR I=1 TO 15:INPUT C(I):NEXT I
FOR I=1 TO 10:INPUT A(I):NEXT I
```

- бир өлчөмдүү массивдин элементтерине формула боюнча маани берүү: `FOR I=1 TO 20:B(I)=5*i*i+4*i-7:NEXT I`

2. Бир өлчөмдүү массивдин маанилерин экранга чыгарууда да цикл уюштурабыз: `FOR I=1 TO 20:PRINT B(I);:NEXT I`

   Эгерде ";" белгисин алып таштасак, анда массивдин элементтери мамыча боюнча чыгат, андай болбогон учурда массивдин элементтери бир сапка жайланышат. Ошондой эле бир эле цикл уюштуруу менен бир нече массивди да чыгарууга болот.

3. Бир өлчөмдүү массивдин элементтеринин суммасын эсептөө

```
S=0:FOR I=1 TO 20:S=S+B(I):NEXT I
```

4. Берилген шартты канааттандырган бир өлчөмдүү массивдин элементтеринин санын эсептөө.

Мейли,бир өлчөмдүү массивдин терс элементтеринин санын эсептөө талап кылынсын.

```
K=0:FOR I=1 TO 5:IF B(I)<0 THEN K=K+1
NEXT I:PRINT "элементтердин саны=";K
```

5. Бир өлчөмдүү массивдин максималдык(минималдык) элементин жана аның катар номерин аныктоо

```
FOR I=1 to 20:INPUT B(I):NEXT I
MAX=B(1):K=0:FOR I=2 to 20 do:IF B(I)<MAX THEN 10
MAX=B(I):K=K+1
10 NEXT I
PRINT "максималдык элемент=";MAX,"катар номери=";K
```

ЭКИ ӨЛЧӨМДҮҮ МАССИВ

Эки өлчөмдүү массивдерди баяндоо үчүн DIM оператору колдонулат. Жалпылуу форматы: `DIM B(N,M); K(3,6); D1(Z,Y)` Бул массивдерди колдонуп маселелерди чыгарууда кийиштирилген циклдар уюштурулат.

Сапча боюнча жылдыруу:

FOR I=1 TO M сыртқы цикл, сапчанын номери өзгөрөт

```
FOR J=1 TO N          ички цикл, мамычанын номери өзгөрөт
...
Next J,I
```

Мамыча боюнча жылдыруу:

```
FOR J=1 TO N сырткы цикл, мамычанын номери өзгөрөт
FOR I=1 TO M ички цикл, сапчанын номери өзгөрөт
...
Next I,J
```

Эки өлчөмдүү массивдерди иштетүүгө арналган базалык алгоритмдерди санап өтөлү:

1. Эки олчомдуу массивдин элементтерине маани берүү
2. Таблица түрүндө чыгаруу
3. Ар бир сапчанын жана мамычанын элементтеринин суммасын эсептоо
4. Массивдин элементтеринин суммасын эсептөө
5. Ар бир сапчанын(мамычанын) максималдуу(минималдуу) элементин жана алардын индекстерин табуу
6. Массивдин максималдуу(минималдуу) элементин табуу

1. Эки олчөмдүү массивдин элементтерине маани берүү.
   Массив 3 сапчадан жана 4 мамычадан б.а. 3x4=12 элементтен турат.
   I сапчанын элементтерине мааниперди берели:

```
FOR I=1 TO 3 DO:FOR J=1 TO 4
INPUT B(I,J):NEXT J:NEXT I
```

   I=1(саптын индекси) учурда J(мамыча) индексинин мааниси 1 ден 4 ко чейин озгөрөт, б.а. сырткы циклдин бир мааниси үчүн ички циклдын бардык маанилери иштетилет да, сырткы циклдын мааниси 1 ге чоңоёт. Көрүнүп тургандай сырткы цикл сапча боюнча, ал эми ички цикл мамыча боюнча уюштурулат. Ошондой эле READ-DATA операторлорунун жардамы менен да массивге маани берсек болот.
2. Берилген эки өлчөмдүү массивдин элементтерин таблица формасында чыгаруу.

```
FOR I=1 TO 3:FOR J=1 TO 4
PRINT B(I,J);:NEXT J:PRINT:NEXT I
```

3. Ар бир сапчанын элементтеринин суммасын эсептөө

```
FOR I=1 TO 3:S=0
FOR J=1 TO 4:S=S+B(I,J)
NEXT J:PRINT I,S:NEXT I
```

4. Массивдин элементтеринин суммасын эсептөө

```
S=0:FOR I=1 TO 3:FOR J=1 TO 4
S=S+B(I,J):NEXT J:NEXT I:PRINT S
```

5. Ар бир сапчанын(мамычанын) максималдуу(минималдуу) элементин жана алардын индекстерин табуу

а) Алгоритмди максималдык элементтин индексин көрсөтүү менен

иштетели.

```
FOR I=1 TO M:MAX=B(I,1)
INDL=I          'саптын номерин сактоо
INDC=1          '1-мамычанын номерин киргизүү
FOR J=1 TO N
IF B(I,J)<MAX THEN 10
MAX= B(I,J):INDC=J 'J-мамычанын номерин сактоо
10 NEXT  J:PRINT i;"-саптын MAX=";MAX:NEXT I
```

б) минималдуу элементти табуу жана ар бир мамыча үчүн аңын индекстерин көрсөтүү алгоритмин карап көрөлү.

```
REM'мамыча боюнча
FOR J=1 TO N :MIN=B(1,J)
INDL=1          'саптын номерин сактоо
INDC=J          'мамычанын номерин киргизүү
FOR I=1 TO M
IF B(I,J)<MIN THEN 10
MIN=B(I,J):INDL=I     'i-саптын номерин сактоо
10 NEXT I:PRINT j;"-мамычанын MIN=";MIN:NEXT J
```

6. Массивдин максималдуу(минималдуу) элементин жана алардын индексин табуу

```
MIN=B(1,1):INDL=1:INDC=1
FOR I=1 TO M:FOR J=1 TO N:IF  B(I,J)>MIN THEN 10
MIN=B(I,J):INDL=I:INDC=J
10 NEXT J:NEXT I
```

Биз мурда карап кеткендей, эки өлчөмдүү массивдин сапчасынын саны мамычасынын саны менен дал келсе, анда мындай массивди квадраттык матрицалар деп атаганбыз. Анда квадраттык матрицанын үстүнөн жүргүзүлгөн кээ бир маселелерге кайрылалы (биз бул маселелер менен Турбо-Паскаль бөлүмүнөн таанышканбыз)

1-Мисал. Башкы диагоналдык элементтердин суммасын эсептегиле.

```
S=0:FOR I=1 TO N:FOR J=1 TO N
IF I=J THEN S=S+B(I,J)
NEXT J:NEXT I
```

2-Мисал. Тескери диагоналынын минималдык элементин тапкыла.

```
MIN=B(1,N)
FOR I=1 TO N
IF B(I,N+1-I)<MIN THEN MIN=B(I,N+1-I)
NEXT I
```

3-Мисал. Матрицаны транспонирлегиле

```
FOR I=1 TO N:FOR J=1 TO M
A(I,J)=B(J,I)
NEXT J:NEXT I
```

3. БЕЙСИК ТИЛИНИН ФУНКЦИЯЛАРЫ

САПТЫК ЧОҢДУКТАР

Сандар менен иштеген сыяктуу эле компьютер текстерди иштетип ага анализ жүргүзө алат. Тексттер символдук өзгөрмө жана массивдер менен берилет, ошондой эле символдор массивдердин элементтери сыяктуу эле жайгашат. Орду(позициясы) номери боюнча иреттелип көрсөтүлүп, кийинки символдорго өтүүдө позициясы бирге чоңоет. Текст менен иштөөдө позициянын номерин бирге озгөртүү менен, бир же ар кандай узундуктагы бир нече символдорду бирге кароого болот. Символдук өзгөрмөлөрдүн сабы каралгандыктан, саптык өзгөрмө же саптык чоңдуктар деп каралат. Саптык чоңдуктарды берүүдө алардын ысмы менен катар "$" белгисин коюн жазышат. Ал эми саптык чоңдукка тиешелүү болгон берилиш тырмакчага алынып жазылат.

Мисалы: `A$="Информатика", B$="Бишкек", G$="Мектеп"`

Саптык чоңдуктар менен ар кандай амалдарды жүргүзүүгө болот.

1. Саптарды бириктирүү операциясы(конкетанация). Бул операция "+" (бул математикадагы кошуу белгиси эмес) белгиси менен белгиленет.

```
A$="китеп":B$="кана":C$=A$+B$
```

жыйынтыгы: `C$="китепкана"`

2. Салыштыруу операциялары: =; >=; >; <>; <; <=. Ар кандай узундуктагы саптарды салыштырууга болот. Салыштыруу ASCII кодуна ылайык солдон онду көздөй жүргүзүлөт.

Ошондой эле саптык чоңдуктарды өзгөртүүгө болот. Ал үчүн атайы функциялар кабыл алынган. Бейсикте символдук (саптык) чоңдуктар менен иштөө функциялары саптык функциялар деп аталат. Саптык функцияларда "$" белгиси коюлуп, символдук чоңдуктар тырмакчага алынып жазылат. Бул функциялардын иштеши менен бир сөздөн башка сөздү бөлүп алууга болот.

- Саптык чоңдуктардын узундугун LEN функциясынын жардамы менен көрсөтсөк болот. Жазылуу форматы төмөнкүдөй:

```
LEN(<саптык туюнтма$>)
```

Мисалы:

```
A$="информатика":K=LEN(A$):PRINT "K=";K
```

Жыйынтыгы: `K=11`

- MID$ оператору саптын көрсөтүлгөн бөлүгүнөн баштап белгиленген узундуктагы символдорду жаңы сапка жайгаштырат. Жашы жазылыш форматы:

```
MID$(<саптык туюнтма$>,N,M)
MID$(<саптык өзгөрмө$>,N,M)=<саптык туюнтма>
```

<саптык туюнтма$> – көрсөтүлгөн сап, N – сапты бөлүп алуучу баштапкы позиция, M– бөлүнүп алына турган саптын узундугу, эгер ал көрсөтүлбөсө, символдорду бөлүп алуу баштапкы позициядан башталат.

1-Мисал:
```
C$="электростанция":B$= MID$(C$,6,4)
PRINT C$,B$
```
C$ өзгөрмөсүнө менчиктелген "электростанция" сөзүнөн, экинчи саптын иштеши менен 6-орундан 4 символ бөлүп алуу менен "рост" сөзү чыгарылат.

2-Мисал:
```
A$="Компьютер - XXI кылымдын техникасы"
B$=MID$(A$,17,5):C$=MID$(A$,26,7)
PRINT A$,B$,C$
```
Бул программанын иштеши менен берилген "Компьютер - XXI кылымдын техникасы" деген саптан жаны "кылым" жана "техника" деген саптар түзүлдү.

3-Мисал: Берилген саптагы "апа" сөзүнүн саныя эсептөө программасын түзөлү.
```
REM'MID
INPUT D$:S=LEN(D$):G=0
FOR I=1 TO S:IF MID(D$,I,3)="апа" THEN G=G+1:
NEXT I:PRINT "апа сөзүнүн саны=";G
END
```
- RIGHT$ оператору саптын оңдогу акыркы позициядан баштап белгиленген узундуктагы символдорду жаңы сапка түшүрөт. Жалпы жазылыш форматы: RIGHT$(<саптык туюнтма$>,M)

  M – бөлүнүп алына турган саптын узундугу

Мисалы:
```
C$="электростанция":A$=RIGHT$(C$,7)
PRINT C$,A$
```
Бул программанын иштөөсүндө берилген "электростанция" сабынын оң тарабынан 7 символ бөлүнүп алыныш, "станция" сабы чыгарылат.

- LEFT$ оператору саптын сол бөлүгүнүн биринчи позициясынан баштап белгиленген узундуктагы символдорду жаңы сапка түшүрөт. Жалпы жазылыш форматы:LEFT$(<саптык туюнтма$>,M)

  M – бөлүнүп алына турган саптын узундугу

Мисалы:
```
C$="электростанция":A$=LEFT$(C$,6)
PRINT C$,A$
```
Бул программанын иштөөсүндө берилген "электростанция" сабынын сол тарабынан 6 символ бөлүнүп алынын, "электр" сабы чыгарылат.

- LCASE$ функциясынын иштеши менен маалыматтарды кичине тамгадан чоң тамгага өтүү аракети аткарылат;

■ UCASE$ функциясы иштеши менен чоң тамгадан кичине тамгаларга өтүү аракети аткарылат.

Мисалы:

```
A$="бакыттын баасы кымбат, бакытка жетиш урмат"
X$= "БИР ТУУГАНДАР"
B$=MID$(A$,16,6):C$=RIGHT$(A$,5):D$=LEFT$(A$,5)
PRINT UCASE$(B$);", ";UCASE$(C$);", ";UCASE$(D$)
PRINT LCASE$(X$)
```

Жыйынтыгында: "КЫМБАТ, УРМАТ, БАКЫТ бир туугандар" сабы чыгат.

■ LTRIM$(<саптык туюнтма$>) функциясы – саптын башындагы пробелдерди өчүрөт

■ RTRIM$(<саптык туюнтма$>) функциясы – саптын аягындагы пробелдерди өчүрөт

Мисалы:

```
A$="    Кыргызстан    "
PRINT "*"+A$+"*":PRINT "*"+LTRIM$(a$)+"*"
PRINT "*"+RTRIM$(a$)+"*"
```

Жыйынтыгы:

```
*    Кыргызстан    *
*Кыргызстан    *
*    Кыргызстан*
```

■ INSTR функциясы – белгиленген саптын бөлүгүн көрсөтүлгөн саптан издөөнү камсыз кылат жана ал саптын башталыш позициясын аныктайт. Жазылуу форматы:

INSTR (M,<саптык туюнтма1>,<саптык туюнтма2> сабы)

M - белгиленген саптын бөлүгүнө алгачкы кирүү позициясы. Эгер M берилбесе, анда издөө 1-позициядан башталат. <саптык туюнтма1> – көрсөтүлгөн сап; <саптык туюнтма2> –белгиленген изделүүчү сап.

Мисалы:

```
A$="Бишкек - кооз шаар":S=INSTR(1,A$,"кооз")
PRINT "Саптын позициясы ";S
```

■ SPACE$(n%) функциясы сапка керектүү сандагы пробелдерди коюуну ишке ашырат. n% – саптагы керектүү болгон пробелдердин саны

Мисалы: FOR i%=1 TO 5:x$=SPACE$(i%):PRINT x$;i%:NEXT i%

■ STR$(n) функциясы сан чөндуктарын литердик чөндүккө айландыруу функциясы.

■ VAL(<саптык туюнтма>) функциясы жазылышында кандайдыр санды камтыган саптык көрүнүштү санга айландыруу функциясы. Эгерде сапта сан көрсөтүлбөсө, жыйынтыгы нөлгө барабар болот.

Мисалы:

```
PRINT "65 ондук сан он алтылык системада "
PRINT "8H+LTRIM$(STR$(41))
PRINT VAL(RIGHT$("УКГ-2001",4)
```

■ STRING$(N,<саптык туюнтма>) функциясы кайталануучу символдордон турган көрсөтүлгөн узундуктагы сапты берет. N –

саптын узундугу. STRING$ функциясы саптын <саптык туюнтманын> биринчи символу менен толтурат.

Мисалы:
```
PRINT STRING$(10, "*");
PRINT "Улуттук компьютердик гимназия";
PRINT STRING$(10, "*")
```
Жыйынтыгы:
```
**********Улуттук компьютердик гимназия**********
```

- ASC(<саптык туюнтма$>) функциясы саптык туюнтмадагы биринчи символду ASCII кодуна айландырат.
- CHR$(ascii-код%) функциясы көрсөтүлгөн ASCII кодго туура келген символду калыбына алып келет.

Мисалы:
```
PRINT ASC("Q")     'Жыйынтык:  81
PRINT CHR$(65)     'Жыйынтык:  A
```

ФУНКЦИЯЛАР

Жогоруда көрсөтүлгөн саптык функциялардан сырткары Бейсик тилинде ар биринин өз милдети болгон функциялар колдонуучунун ишин жеңилдетет. Алардын кээ бирлери менен таанышалы.

- SGN(<сан туюнтмасы> функциясы сан туюнтмасынын белгисин аныктайт. Эгер туюнтма оң сан болсо, анда ал 1 ге, эгер туюнтма терс сан болсо, анда ал -1 ге, эгер нөлгө барабар болсо, анда ал 0 гө барабар болот.

Мисалы: PRINT SGN(-345), SGN(451), SGN(0)

Жыйынтыгы: -1 1 0

- CDBL(<сан туюнтмасы>) функциясы сан туюнтмасын экилик тактыктагы маанiге айлантат.
- CSNG(<сан туюнтмасы>) функциясы сан туюнтмасын бирдик тактыктагы маанiге айлантат.

Мисалы: PRINT 2/3,CDBL(2/3)

Жыйынтыгы: .6666667 .6666666666666666

PRINT CSNG(9565.674915150001#) 'Жыйынтыгы: 9565.675

- CINT(<сан туюнтмасы>) функциясы сан туютмасынын маанисин бүтүн маанiге чейин тегеректейт. Сан туюнтмасынын мааниси – 32768 ден 32767 ге чейинки аралыкта болушу зарыл.
- CLNG(<сан туюнтмасы>) функциясы сан туюнтмасын узун бүтүн маанiге(4 байт) чейин тегеректейт. Сан туюнтмасынын мааниси – 2147483648 ден 2147483647 ге чейинки аралыкта болушу зарыл.

Мисалы: PRINT CINT(36.2459), CINT(562.891)

Жыйынтыгы: 36 563

PRINT CLNG(338457.859#) 'Жыйынтыгы: 338458

- FIX(<сан туюнтмасы>) функциясы жылыш жүрүүчү үтүрдү камтыган туюнтманы бүтүн бөлүгүнө чейин тегеректейт.
- INT(<сан туюнтмасы>) функциясы сан туюнтмасынын андан кичине же ага барабар болгон эң чоң бүтүн бөлүгүн көрсөтөт.

Мисалы: PRINT FIX(12.9999), FIX(-13.04)
PRINT INT(12.9999), INT(-13.04)
Жыйынтыгы: 12 -13
12 -14

- HEX$(<сан туюнтмасы>) санды он алтылык эсептөө системасында көрсөтөт.
- OCT$(<сан туюнтмасы>) санды сегиздик эсептөө системасында көрсөтөт. Сан туюнтмасы бүтүнгө чейин тегеректелет.

Мисалы:
```
INPUT x
A$ = HEX$(x):B$ = OCT$(x)
PRINT X; "16 лык эсептjj системасында";A$;
PRINT X;"8  дик эсептjj системасында";B$
```

- RANDOMIZE оператору капилет сандар генераторун орнотот. Ал эми RND(n#) бирдик тактыктагы, 0 жана 1 дин ортосундагы капилет санды берет.

Мисалы: RANDOMIZE TIMER
```
x%=INT(RND*6)+1:y%=INT(RND*6)+1
PRINT "экрандын координаталары:x=";x%;"; жана y=";y%
```

- SPC(n%) функциясы PRINT же LPRINT операторлорунда берилген сандагы пробелдерди калтырат. n% – мааниси 0 дөн 32767 ге чейин болгон пробелдердин саны.

Мисалы: PRINT "Мектеп"; SPC(5); "Китеп"
Жыйынтыгы: Мектеп Китеп

КОЛДОНУУЧУ ҮЧҮН ФУНКЦИЯЛАР.

Колдонуучу үчүн функциялардын аныкталышы стандарттык функциялардан башкача. Колдонуучу үчүн функциялар DEF FN оператору менен берилет. Бул функциянын жазылыш форматы:

```
DEF FN<ысмы> = <туюнтма>
```

Жалпы көрүнүшү:

```
DEF FN<ысмы>
<операторлор тобу>
FN<ысмы> = <туюнтма>
<операторлор тобу>
EXIT DEF
<операторлор тобу>
END DEF
```

Формалдуу параметр катары каралган өзгөрмөнүн аты локалдуу болуп эсептелет б.а. функциянын баяндалышынан башка өзгөрмөлөр үчүн колдонулушу мүмкүн. Туюнтмага башка өзгөрмөлөр да кирги зилиши мүмкүн. Бул өзгөрмөлөрдүн маанилери глобалдуу болуп эсептелет, б.а. программанын калган бөлүктөрүндө ошол эле ысым менен колдонулат.

Функция саптык же сан чоңдуктары менен аныкталышы мүмкүн. Стандарттык эмес тригонометриялык функциялардын жа- зылышын карайлы.

Функциялардын жазылышы	Функцияга кайрылуу
DEF FNT(X)=COS(X)/SIN(X)	FNT(X)
DEF FNS(X)=1/COS(X)	FNS(X)
DEE FNC(X)=1/SIN(X)	FNC(X)
DEF FNH(X)=(EXP(X)-EXP*(-X))/2	FNH(X)
DEF FNQ(X)=(EXP(X)-EXP*(-X))/2	FNQ(X)
DEF FNA(X)=2*((1-SQR(1-X^2))/X	FNA(X)
DEF FNL(X) = LOG(X/LOG(10))	FNL(X)

Колдонуучу үчүн функцияларды колдонуп өзгөрмөлөрдүн типтерин баяндоого болот.

DEF INT<аталыштын алгачкы тамгасы> – бүтүн типтер үчүн

DEF SNG<аталыштын алгачкы тамгасы> – анык бирдик тактыктагы типтер үчүн

DEF DBL<аталыштын алгачкы тамгасы> – анык экилик тактыктагы типтер үчүн

DEF STR<аталыштын алгачкы тамгасы> – саптык типтер үчүн

Мисалы: DEF INT A,B:DEF DBL X:DEF STR S

Мында берилген аталыштар: A,B – бүтүн типте; X – экилик тактыктагы анык типте; S – саптык типте экендиги көрсөтүлөт

1-Мисал: эки чекиттин арасындагы аралыкты табуу программасын түзүү мисалын карайлы.

```
DEF SNG X,Y,Z:READ X1,X2,Y1,Y2
Z=SQR((X1-X2)^2+(Y1-Y2)^2)
PRINT "аралык Z="; Z: DATA 4,8,3,9
```

2-Мисал: натуралдык сандардын квадраттары жана кубдарын табуу программасын түзүү.

```
DEFINT K-N: FOR K=1 TO 30:N=K*K: M=N*K
PRINT K; TAB (6);N;TAB(12);M:NEXT K
```

3-Мисал:

```
INPUT C,D:DEF FNR(X,Y,Z)=COS(X-Y-Z)
T=TAN(FNR(C,D,22)+FNR(41,C-D,ABS(78)))
PRINT T:END
```

КАМТЫЛГАН ПРОГРАММАЛАР

Бейсикте программалар бүтүн катары кабыл алынып, ушул эле программада атайы эрежелер менен уюштурулган программалык саптарга кайрылуу учурлары кездешет. Программанын баштапкы бөлүгүн негизги программа, кайрылуу мүмкүндүгүндө түзүлгөн бөлүгү камтылган программа деп атайбыз. Камтылган программа программанын негизги бөлүгүн аяктоо END операторунан кийин жазылганы ыңгайлуу. Камтылган программага кайрылуу операторунун жазылыш форматы: GOSUB N

Бул оператордун аткарылышы менен N саптан башталган камтылган программага башкаруу өтөт да, камтылган программаны аяктоочу RETURN операторуна чейинки саптар аткарылат. Камтылган программа аткарылгандан кийин башкаруу GOSUB дан кийинки операторго өтөт. Бир программада бир нече камтылган программа түзүүгө, ошондой эле камтылган программага бир нече жолу кайрылууга болот. Камтылган программанын функциядан айырмасы формалдуу параметри жолукпайт, ошондуктан локалдуу өзгөрмөлөр жок. Бардык өзгөрмөлөр программада колдонулгандай эле камтылган программада да колдонулат. Эгер программада өзгөртүлсө, анда камтылган программада өзгөртүлүшү боюнча колдонулат.

1-Мисал:

```
A=2:B=3.5:C=7:GOSUB 100
A=2:B=5:C=8.7:GOSUB 100:END
100 X=(A+B)*(B-C)^2
PRINT X: RETURN
```

2-Мисал: a,b,c жактары боюнча үч бурчтуктун аянтын табуу

```
INPUT "Үч бурчтуктун жактарын киргиз";a,в,c
GOSUB 10
PRINT "Үч бурчтуктун аянты S="; S:END
10 REM' Үч бурчтуктун аянтын табуу
P=(A+B+C)/2:S=P*(P-A)(P-B)(P=C):RETURN
```

3-Мисал: Y=2x+sinx жана Y=cosx-x/2 функцияларынын маанилерин [-3,3] интервалда 0.5 кадамы менен эсептөө программасын түзгүлө

```
REM функциянын маанисин эсептөө
DEF FNY=2*X+SIN(X)
PRINT "Y=2*X+SIN(X) Функциясын эсепте":GOSUB 10
DEF FNX=COS(X)-X/2
PRINT "Y=cosx-x/2":GOSUB 10:END
10 REM камтылган программа
PRINT TAB(5);"U";TAB(33);"P"
FOR X = -3 TO 3 STEP .5
U=FNX:P=FNY:PRINT U,TAB(15);,P:NEXT X
```

4-Мисал: маек программасы

```
PRINT "жадыбал билесизби"
INPUT "кайсы санды көбөйтөлү"; A,B
PRINT A; "жерде"; B; "канча":GOSUB 10:END
10 REM 'Камтылган  программа
15 INPUT "жооп"; C:IF C <> A*B GOTO 20
PRINT "Бали,азамат,туура!!!": GOTO 30
20 PRINT "Туура эмес,кайра жооп бер!!!":GOTO 15
30 RETURN
```

ON туюнтманын маанисине жараша бир же бир нече абалдардын бирине өтүүдө ON GOSUB операторлору да колдонулат. ON GOSUB операторунун иштеши ON GOTO операторунун иштеши сыяктуу эле. Бул операторлордун жазылыш форматы:

```
ON <туюнтма> GOSUB <сан тизмеги>
```

Мында: <туюнтма> 0 дөн 250 гө чейинки маани алат (сап боюнча символдордун саны). <сан тизмеги> – сап номери же белги (метка) тобу. Мисалы:

```
FOR I%=1 TO 2:ON I% GOSUB One,Two:NEXT I%:END
One: PRINT "бир":RETURN
Two: PRINT "эки":RETURN
```

TIMER оператору таймер үчүн учурдагы окуяны ылгалоону иштетет, убактылуу токтотот, иштетпей алып салат. Учурдагы ылгалоону иштетүүдө ON TIMER оператору көрсөтүлгөн секунданын саны өткөндөн кийин камтылган программага кайрылууну камсыз кылат.

TIMER ON – таймер үчүн окуяны ылгалоону иштетет.

TIMER OFF – таймер үчүн окуяны ылгалоону иштетпейт.

TIMER STOP – ылгалоону убактылуу токтотот, TIMER ON дун иштеши менен улантылат

```
ON TIMER(n%) GOSUB <сап>
```

n% – окуяны ылгалоо камтылган программасына ON TIMER кайрылуу жасаганга чейинки секундалардын саны, ал 1 ден 86400 гө чейинки маани алат (24 саат).

<сап> – окуяны ылгалоочу камтылган программанын биринчи сабынын номери же меткасы. Мисалы:

```
ON TIMER(1) GOSUB Time
TIMER ON:CLS:PRINT "Убакыт: ";:S=TIMER
WHILE A<10:A=TIMER-S:WEND
END
Time:LOCATE 1,8:PRINT TIME$:PRINT "АЙДА":RETURN
```

Бул мисалда каралып кеткен TIME$ оператору компьютерге учурдагы системалык убакытты орнотот: TIME$=<саптык туюнтма$>

<саптык туюнтма$> – төмөндө көрсөтүлгөн форматтардын биринде көрсөтүлгөн убакыт:

чч – саатты орнотуу, минута жана секунда бул учурда нөлгө барабар.

чч:мм – саатты жана минутаны орнотуу, бул учурда секунда нөлгө барабар. чч:мм:сс – саатты, минутаны жана секунданы орнотуу.

Ал эми TIME$ функциясы сапты чч:мм:сс форматына алып келет.

DATE$=<саптык туюнтма> оператору компьютердин учурдагы системалык датасын көрсөтөт. <саптык туюнтма> – төмөндөгү форматтардын бирин алат: аа/кк/жж, аа–кк–жж, аа:кк:жж,

Мисалы: PRINT DATE$:DATE$ = "07-21-99"
PRINT"көрсөтүлгөн дата";DATE$

ФАЙЛДАР МЕНЕН ИШТӨӨ

Компьютерде маселе чыгарууда чоң өлчөмдөгү берилиштердин жыйындысын колдонуу учурлары кездешет. Бул берилиштерди керектүү өлчөмдө колдонууга(алууга);

- бир эле маалыматтарды бир нече программаларда колдонууга;
- кайрадан аталышы менен эске сактап колдонууга болот.

Аталышы менен берилген маалыматтардын жыйындысы файл деп аталат. Файлдардын массивден айырмасы жалаң гана бир типтүү элементтер эмес, ар түрдүү типтеги элементтерди камтыганында.

Бейсикте файлдар менен иштөөдө:

- аларды түзүү;
- андагы маалыматтарды колдонуп, эске сактоо;
- ага кайрадан кайрылуу, аны өзгөртүү

сыяктуу аракеттерди жүргүзөбүз. Программа менен иштөөдө, программа ОЗУга жазылып, файлдарга кайрылууда сырткы түзүлүштөр менен байланыш жасалат. Бул учурда файлдын ысмын бир каналдын номери менен байланыштырып, OPEN оператору менен ачык деп жарыялоо керек. Файлды ачык деп жарыялагандан кийин ага маалымат жазуу же окуу жүргүзүлөт. Эгер берилиш(ОЗУга) сырттан окулса INPUT, сыртка чыгарылыш керек болсо OUTPUT оператору колдонулат. Берилиштер бул жолдор менен алынып колдонулгандан кийин PRINT(WRITE) оператору менен тиешелүү түзүлүштөргө чыгарылат. Файлдар менен иштөө аракеттери аяктагандан кийин, берилиштерди киргизүү жана чыгаруу CLOSE оператору менен жабылат.

Файлды ачууда OPEN оператору колдонулаарын жогоруда айтып оттук. Жалпы жазылыш формасы төмөнкүдөй:

OPEN <файл$> FOR <режим> ACCESS <жол> <жабуу> AS #<файлдын номери> LEN =<жазылыштын узундугу>

<файл$> – файлдын ысмы. Ал түзүлүштү же файлга баруу жолун ким

тышы мүмкүн. <режим> - APPEND,BINARY,INPUT,OUTPUT, RANDOM режимдеринин бири. APPEND оператору файлды удаалаш чыгарууну жана файлдын көрсөткүчүн файлдын аягына коюну ишке ашырат. Бул учурда PRINT # же WRITE # операторлору файлды толукташат. BINARY оператору файлдын бинардык режимде иштөөсүн камсыз кылат. Бинардык режимде GET жана PUT операторлорун колдонуп информацияны каалаган байттын позициясынан алып салуу же жазууну ишке ашырууга болот. INPUT оператору файл удаалаш киргизүү үчүн ачылгандыгын көрсөтөт. OUTPUT оператору файл удаалаш чыгаруу үчүн ачылгандыгын көрсөтөт. RANDOM оператору файлга түз аракет жасоо режими үчүн файлды ачууну камсыз кылат. ACCESS <жол> - тармак менен иштөөдө файлдын окуу(READ) же жазуу (WRITE) же окуу-жазуу(READ WRITE) үчүн ачылгандыгын көрсөтөт. Жазылуу форматы: ACCESS <READ же WRITE же READ WRITE>. READ WRITE бинардык жана түз аракет жасоо режимдери, ошондой эле APPEND менен ачылган файлдар үчүн гана аракетте болот. <жабуу> AS - тармактык түз аракет үчүн файлдын жабылуу шартын аныктайт: SHARED(жалпы), LOCK READ (окуу үчүн жабуу), LOCK WRITE(жазуу үчүн жабуу), LOCK READ WRITE(окуу-жазуу үчүн жабуу), <файлдын номери> - ачылган файлды идентификациялоочу 1 ден 255 ке чейинки сан. LEN =<жазылыштын узундугу> - түз аракеттеги файлдар үчүн жазылыштын узундугу(128 байт); удаалаш файлдар үчүн буфердеги символдордун саны(512 байт)

1-Мисал:
```
OPEN "TEST.DAT" FOR OUTPUT AS #1
PRINT #1, "Текст":CLOSE
OPEN "TEST.DAT" FOR INPUT AS #1
PRINT INPUT$(3,1):CLOSE
```

2-Мисал:
```
INPUT "файлдын атын киргиз:";n$
OPEN n$ FOR OUTPUT AS #1
PRINT #1,"бул файлда сакталат":CLOSE
OPEN n$ FOR INPUT AS #1
INPUT #1,a$:PRINT "файлдан окуу:"; a$:CLOSE
```

3-Мисал: Томондогу программада окуучулардын информатика сабагы боюнча жетишүүсүнүн жыйынтыгы файл формасында көрсөтүлдү.
```
CLS:B$ = "##### \                    \####"
REM 'БЕРИЛИШТЕРДИ  ДИСККЕ ЖАЗУУ
OPEN "9А-КЛАСС" FOR OUTPUT AS #1
FOR I%=1 TO 10:PRINT I%;"-окуучунун фамилиясын киргиз"
PRINT "Оозсын киргиз":INPUT FIO$,BAA%
WRITE #1,I%,FIO$,BAA%:NEXT I%
CLOSE #1:CLS:PRINT STRING$(10,"*");
PRINT "УЛУТТУК КОМПЬЮТЕРДИК ГИМНАЗИЯ";
```

```
PRINT STRING$(10,"*"):PRINT
PRINT "ИНФОРМАТИКА САБАГЫ БОЮНЧА ЖЫЙЫНТЫК БААЛАРЫ"
PRINT STRING$(15, "*");:PRINT "9А КЛАСС";
PRINT STRING$(15, "*"):PRINT STRING$(50,"-")
A$ = "n/n  ФАМИЛИЯСЫ,АТЫ          ИНФОРМАТИКА"
PRINT A$:PRINT STRING$(50,"-")
REM 'Берилиштерди иштетуу
OPEN "9А-КЛАСС" FOR INPUT AS #1
FOR I%=1 TO 10:INPUT #1,NOM%,FIO$,BAA%
PRINT USING B$;NOM%;FIO$;BAA%
NEXT I%:PRINT STRING$(50,"-")
END
```

Берилиштерди(фамилия, баасы) киргизүү менен бул жыйынтыкты алабыз:

```
**********УЛУТТУК КОМПЬЮТЕРДИК ГИМНАЗИЯ**********
    ИНФОРМАТИКА САБАГЫ БОЮНЧА ЖЫЙЫНТЫК БААЛАРЫ
     **************9А КЛАСС**************
--------------------------------------------------
n/n  ФАМИЛИЯСЫ,АТЫ          ИНФОРМАТИКА
--------------------------------------------------
    1 Аскеев Канат                4
    2 Азыкулов Бактияр            5
    3 Жапарова Элизат             5
    4 Жаманкулов Кайрат           4
    5 Каракинов Адилет            4
    6 Саидов Таалай               5
    7 Мамаюсупов Акылбек          4
    8 Шайымкулов Таалай           5
    9 Чалакеев Дастан             5
   10 Айнакулова Мээрим           4
--------------------------------------------------
```

SLEEP <секунда> **оператору программанын аткарылышын кандайдыр бир** <секунда> **убакытка токтотот. Эгер секунда 0 гө барабар болсо же жазылбаса, анда программа клавиша басылгыча токтойт. Мисалы:**

```
PRINT "10 сек га үргүлөгүлө?":SLEEP 10
PRINT "ойгонгула!"
```

CHAIN <файл$> – **учурдагы программадагы башкарууну башка** <файл$> **программага берүү. Мисалы:** TEST.BAS **файлы** \DOS **каталогунда жайгашкан деп эсептесек, аны мындай жазууга болот:**

```
CHAIN "C:\DOS\TEST.BAS"
```

CHDIR <жол$> – **каталогдун түзүлүшүн өзгөртүү.** <жол$> – **жаңы багытталган каталогдун багыты.** CLEAR – **баардык файлдарды жабат, файлдар буферин тазалайт, жалпы өзгөрмөлөр жана массивдин элементтерин нөлгө барабарлайт.** CLOSE #<файлдын номери>, ...

бир же бир нече файлдар же түзүлүштөрдү жабуу. #<файлдын номери> – учурдагы файл же түзүлүштүн номери. CLOSE аргументсиз учурдагы баардык файл жана түзүлүштөрдү жабат.

ТЕКСТТЕР МЕНЕН ИШТӨӨ

Бейсик тилинде тексттер менен иштөөдө төмөнкү операторлорду колдонобуз: CLS – (Clear Screen – экранды тазалоо) текстик же графикалык экранды тазалоо. Графикалык сүрөттөлүштү жок кылып, жаңы сүрөттөлүштү чыгарууда бир же бир нече кадрларды алмаштырууда колдонулат. Ар бир жаңы сүрөттөлүш(кадр) CLS менен башталышы керек.

CLS 0 – толук экранды тазалоо

CLS 1 – графикалык экранды же толук экранды тазалайт

CLS 2 – текстик аймакты тазалайт

LOCATE оператору – экранда курсордун позициясын жылдырууну камсыз кылат. Жазылуу форматы:

LOCETE <сап%>,<мамыча%>,<курсор%>,<старт%>,<стоп%>

<сап>,<мамыча> – курсордун сап же мамыча боюнча жылуусунун номери. <курсор>– курсордун аныктальшы(0–көрүнүүчү, 1–көрүнбөөчү). <старт>,<стоп> – берилүүчү сапчанын баштапкы жана акыркы маанисин(0 дөн 31 ге чейин) аныктоо.

SCRLIN оператору – курсордун сапча боюнча баштапкы маанисине кайрылууну камсыз кылат.

POS<туюнтма> – курсордун мамыча боюнча учурдагы абалын аныктайт.Мисалы:

```
CLS:LOCATE 7,23:MyR%=CSRLIN:MyC%=POS(0)
PRINT "АЛМАЗ (Каалаган баскычты баскыла)"
DO: LOOP WHILE INKEY$ = "":LOCATE (MyR%+6),(MyC%+5)
PRINT "АЗИЗ ЖАНА АЛМАЗ - ДОСТОР"
```

БЕЙСИКТИН ҮН ЧЫГАРУУ МҮМКҮНЧҮЛҮГҮ

Бейсиктин негизги мүмкүнчүлүктөрүнүн бири үн чыгаруу болуп эсептелет. Ал үчүн төмөндөгү операторлор колдонулат:

BEEP – компьютердин динамиги аркылуу үн сигналын чыгаруу.

PLAY <командалар сабы$> – музыка чыгарууну башкарат(үн берет, очуроt, убакытты токтотот),<командалар сабы$> – төмөнкү бир же бир нече командалардан турган туюнтма:

октава жана тондун командалары

O<октава> – учурдагы октаваны көрсөтөт (0 - 6). <or> – бир октавага жогору же төмөн отүү A – G – учурдагы октаванын белгилүү нота-

сын ойнотот. N<нота> – жети октава диапозонундагы (0 -84) аныкталган нотаны ойнотот.

Узактык жана темптин командалары:

L<**өлчөм**> – Ар бир нотанын(1 - 64) узактыгын берет. L1 – бүтүн нота, L2 – нотанын 1/2 бөлүгү ж.б. ML – legato аткаруу түрү. MN – normal аткаруу түрү. MS – staccato аткаруу түрү. P<пауза> – паузаны(1 - 64) берет. P1 - бүтүн нотадагы пауза, P2 - 1/2 нотадагы пауза ж.б. T<темп> – чейрек минутада(32-255) аткаруу темпин берет.

Режим командалары: MF – негизги үн чыгаруу, MB – фону менен үн чыгаруу. **Нотаны өзгөртүү командалары**: "#" же "+" – диез; "–" –Бемоль; "." – нотанын өлчөмүнө карата болгон 3/2 узактык.

PLAY командалар сапчасынан PLAY сапчанын бөлүгүнүн командаларын аткаруу үчүн "X" командасын колдонобуз:

```
PLAY "X"+ VARPTR$(командалар сабы$)
```

Мисалы:

```
scale$="CDEFGAB":PLAY "L16": FOR i%=0 TO 6
PLAY "O"+STR$(i%):PLAY "X"+VARPTR$(scale$):NEXT i%
```

Иштеп жаткан учурда ON PLAY оператору качан гана музыкалык буфердеги нотолардын саны көрсөтүлгөн сандан кичине болгон учурда камтылган программага кайрылып турат.

PLAY ON – музыка ойношунда кубулуш жүрүшүн иштетет.

PLAY OFF – музыка ойнолушунда кубулуш жүрүшүн токтотот.

PLAY STOP – музыка ойнолушун убактылуу токтотот.

ON PLAY(<кезек **күтүү** лимити%>) GOSUB <сап>

<кезек **күтүү** лимити%>) – 1 ден 32 ге чейинки сандар.

<сап> – камтылган программанын биринчи сабынын номери

Мисалы:

```
ON PLAY(3) GOSUB Background:PLAY ON
Music$="Mbo3L8ED+ED+Eo2Bo3DCL2o2A"
PLAY Music$:LOCATE 2,1
PRINT "токтотуу үчүн каалаган баскычты бас";
DO WHILE INKEY$="": LOOP: END
Background:I%=I%+1:LOCATE 1,1
PRINT " фон чакырылды"; I%; "(а)жолу ";
PLAY Music$:RETURN
```

SOUND<жыштык,узактык> оператору – компьютердин динамиги аркылуу үнүн чыгаруу.

<жыштык> – 37 ден 32767 аралыгында Гц менен берилген үндүн жыштыгы. <узактык> – үндүн созулушун көрсөткөн системалык сааттын тактыларынын саны, мааниси 0 дөн 65535 диапазонунда жатат. Ар бир секундада 18,2 такт болуп өтөт.

Мисалы:

```
FOR i%=440 TO 1000 STEP 5:SOUND i%,i%/1000: NEXT i%
```

4. БЕЙСИК ТИЛИНИН ГРАФИКАЛЫК МҮМКҮНЧҮЛҮКТӨРҮ

ГРАФИКАДАГЫ АЛГАЧКЫ КАДАМДАР

Тегиздикте, мейкиндикте фигуралар менен байланышкан анализ, синтез, геометриялык процедуранын аткарылышы, графикалык объектлердин моделдерин баяндоону калыптандыруу жана киргизүүнү уюштуруу ар кандай графиктер, гистограммалар, чиймелердин эскизи жана башка графикалык документтерди чыгаруу менен жүргүзүлөт. Графикалык информацияларды чагылдыруунун негизги каражаттарына токтололу.

Бейсикте сүрөттөлштөрдү чыгаруудан мурун графикалык режимди орнотуу керек. Ал үчүн SCREEN оператору колдонулат. Анын кызматы графикалык режимди жана экрандын башка мүнөздөмөлөрүн орнотот. Бул оператордун жалпы түрдө жазылышы төмөндөгүдөй:

SCREEN<режим%>,<түс%>,<активдүү сап>,<көрүнүүчү сап>

<режим%> – экрандын режимин коет. <түс%> – түстүү жана монохромдуу сүрөттөлүштөрдүн(0 жана 1 режимде гана) ортосундагы түстөрдүн алмашуу маанилери(0 же 1) маанилери. Эгерде режим 0, маани 0 гө барабар болсо, анда түс берилбейт, эгер режим 0, маани нөлдөн башка сан болсо, анда түс берилет. Эгер режим 1, маани 0 болсо, анда түс берилбейт, эгер режим 1, маани нөлдөн башка сан болсо, анда түс берилет. <активдүү сап> – текст же графиктеринин жыйынтыгы чыгарылуучу экрандын бети. <көрүнүүчү сап> – учурдагы чагылдырылган экрандын бети.

Төмөндө экрандын режими тууралуу жалпыланган маалымат берилген:

1. MDPA, CGA, Hercules, Olivetti, EGA, VGA же MCGA адаптерлери үчүн

SCREEN 0 – Тексттик режими

- тексттин форматы: 40x25, 40x43, 40x50, 80x25, 80x43 же 80x50, ал эми символдун форматы: 8x8(8x14, 9x14 же 9x16 EGA же VGA үчүн)
- каалагандай 16 атрибутка(CGA же EGA үчүн) 16 түс туура келет
- каалагандай 16 атрибутка(VGA же EGA үчүн) 64 түс туура келет
- тексттин жана адаптердин мүмкүнчүлүгүнө жараша видеопамяттын өлчөмү 8 бет(0–7), 4 бет(0–3), 2 бет(0–1) же 1 бет(0)

2. CGA, EGA, VGA же MCGA адаптерлери үчүн

SCREEN 1 – 320x200 графикалык режими

- текстгин форматы 40x25, символдун форматы 8x8
- 16 фондуу түс жана CGA үчүн COLOR операторун колдоногондо менчиктелген 3 түстүү алдыңкы пландын жыйындыларынын бири
- 16 түс EGA же VGA үчүн 4 атрибутка берилет
- видеопамяттын 1 бети(0)

SCREEN 2 – 640x200 графикалык режими

- текстгин форматы 80x25, символдун форматы 8x8
- 16 түс EGA же VGA үчүн 2 атрибутка берилет
- видеопамяттын 1 бети(0)

3. Hercules, Olivetti же AT&T адаптерлери үчүн

SCREEN 3 –720x348 графикалык режим; Hercules адаптери жана монохромдуу монитор гана талап кылынат

- текстгин форматы 80x25, с имволдун форматы 9x14
- Көбүнчө видеопамяттын 2 бети(0–1), эгер 2-түстүү видеоадаптер орнотулса, анда 1 бет(0)
- PALETTE оператору иштетилбейт.
- Экрандын 3-режиминде иштөөдөн мурун Hercules дин драйверин чакыруу керек.

SCREEN 4 – 640x400 графикалык режими; Olivetti фирмасынын M24,M240,M28,M280, M380,M380/C жана M380/T компьютерлери, ошондой эле AT&T фирмасынын 6300 сериясындагы персоналдык компьютерлери үчүн аткарылат.

- текстгин форматы 80x25, символдун форматы 8x16
- COLOR оператору аркылуу тандалган алдыңкы пландын түсүно 16 түстүн ичинен бири гана берилет; экрандын фону дайыма кара болот.
- видеопамяттын 1 бети(0)
- PALETTE операторуиштетилбейт.

4. EGA же VGA адаптерлери үчүн

SCREEN 7 – 320x200 графикалык режими

- текстгин форматы 40x25, символдун форматы 8x8
- 16 атрибуттун каалаганына 16 түстөр туура келет.
- 64 К эстеги EGA адаптери үчүн – видеопамяттын 2 бети(0–1), калган учурларда – 8 бет(0–7) туура келет.

SCREEN 8 – 640x200 графикалык режими

- текстгин форматы 80x25, символдун форматы 8x8
- 16 атрибуттун каалаганына 16 түстөр туура келет.
- 64 К эстеги EGA адаптери үчүн – видеопамяттын 1 бети(0), калган учурларда – 4 бет(0–3) туура келет.

SCREEN 9 – 640x350 графикалык режим

- текстгин форматы 80x25 же 80x43,символдун форматы 8x14 же 8x8
- 4 атрибутка(адаптердин эси 64 К) 16 түс туура келет же 16 атрибутка(адаптердин эси 64К дан жогору) 64 түс туура келет.
- 64 К эстеги EGA адаптери үчүн – видеопамяттын 1 бети(0), калган учурларда – 2 бет(0–1) туура келет.

5. EGA же VGA адаптерлери, монохромдуу мониторлор үчүн

SCREEN 10 – 640x350 графикалык режими

- текстгин форматы 80x25 же 80x43,символдун форматы 8x14 же 8x8
- 4атрибутка 9 псевдотүскө чейинки маанилер берилет.
- видеопамяттын 2 бети(0–1), 256К эстеги адаптер керек.

SCREEN 11 (VGA же MCGA) – 640x480 графикалык режими

- текстгин форматы 80x30 же 80x60,символдун форматы 8x16 же 8x8
- 2 атрибутка 256 К эске чейинки түстөрдү берүү
- видеопамяттын 1 бети(0)

SCREEN 12 (VGA) – 640x480 графикалык режими

- текстгин форматы 80x30 же 80x60,символдун форматы 8x16 же 8x8
- 16 атрибутка 256 К эске чейинки түстөрдү берүү
- видеопамяттын 1 бети(0)

SCREEN 13 (VGA же MCGA) – 320x200 графикалык режими

- текстгин форматы 40x25, символдун форматы 8x8
- 256 атрибутка 256 К эске чейинки түстөрдү берүү
- видеопамяттын 1 бети(0)

COLOR – экрандын түсүн берүү. Жазылуу форматы:

```
COLOR <негиз%>,<фон%>,<чек%>, {экрандын режими-0}
COLOR <фон%>,<палитра%>,      {экрандын режими-1}
COLOR <негиз%>                {экрандын режими-4,12,13}
COLOR <негиз%>,<фон%>     {экрандын режими- 7,10}
```

<негиз%> – экрандын негизги түсүн аныктоо, <фон%> – экрандын фондук түсүн аныктоо, <чек%> – экрандын чектик түсүн аныктоо <палитра%> – түстүн атрибутун аныктоо(0 же 1)

палитра%	Атрибут 1	Атрибут2	Атрибут3
0	жашыл	кызыл	күрөң
1	көгүш	кочкул кызыл	ачык ак

Түстүн атрибуттары жана маанилери компьютердин графикалык адаптерине жана акыркы SCREEN оператору менен орнотулган экрандын режимине көз каранды болот. Эгер компьютердин системасында EGA, VGA же MCGA адаптерлери болсо, анда түстүн атрибуттарын

өзгөртүү үчүн PALETTE операторун колдонууга болот. Жазылуу форматы:

```
PALETTE <атрибут%>,<түс&>
PALETTE USING  <массивдин аты#(индекс%)>
```

<атрибут%> – түстүн өзгөрүү атрибуту, <түс&> – атрибутка менчиктелген түстүн мааниси, <массивдин аты#> – экрандын түстөрүнүн учурдагы атрибуттарынын тобуна туура келген түстөрдүн маанилер массиви. Бардык атрибуттарга түстөрдү менчиктөө үчүн массив жетишээрлик чоң болушу керек, <индекс%> – атрибутка менчиктеле турган массивдин биринчи элементинин индекси.

Экрандын графикалык режими үчүн түстөрдү берүү төмөнкү константаларга менчиктелген:

Black(кара)	0	DarkGray (бозомтук)	8
Blue(көк)	1	LightBlue (көгүш)	9
Green(жашыл)	2	LightGreen (ачык жашыл)	10
Cyan(Циан)	3	LightCyan (ачык циан)	11
Red(кызыл)	4	LightRed (ачык-кызыл)	12
Magenta(сыя көк)	5	LightMagenta (ачык-сыя көк)	13
Brown(күрөң)	6	Yellow(сары)	14
LightGray (ачык- боз)	7	While(ак)	15

ЖӨНӨКӨЙ ГРАФИКАЛЫК ФИГУРАЛАР

Жөнөкөй графикалык фигуралар катарында чекит, кесинди, төрт бурчтук, айлана эсептелерин билебиз. Мына ушул фигураларды Бейсиктин графикалык мүмкүнчүлүгүн колдонуп, аларды сызуу аракеттери менен таанышалы.

PSET(x,y),<color> жана PRESET(x,y),<color> операторлору чекитти сызууну ишке ашырат. Мында (x,y)– чекиттин экрандагы координатасы, <color> – чекиттин түсү, эгерде <color> берилбесе, анда PRESET операторунун аткарылышы менен чекит учурдагы экрандын фонунун түсүн алат, ал эми PSET оператору аткарылганда чекит экрандын алдыңкы планынын түсүн алат.

Мисалы:

```
FOR i%=0 TO 320:PSET(i%,100):PRESET(i%,100):NEXT i%
```

Экранда кесиндини сызуу LINE операторунун жардамы менен аткарылат: LINE(x1,y1)-(x2,y2),<түс%>

Мында (x1,y1)– кесиндинин 1-учунун координатасы, (x2,y2) – кесиндинин 2-учунун координатасы, <түс%> – кесиндинин түсү.

Мисалы: LINE(100,150)-(300,250),7

Экранда тик бурчтук сызууну төмөндөгү оператордун жардамы

менен аткарабыз: LINE (X1,Y1)-(X2,Y2),<түс%>,B
Мында: (X1,Y1),(X2,Y2)- тик бурчтуктун диогоналы боюнча баштапкы жана акыркы чекиттеринин координаталары, <түс%> – тик бурчтуктун чегинин түсү. Түстүн атрибуттары SCREEN операторунда көрсөтүлгөн экрандын режиминен жана графикалык адаптерден көз каранды. B – Тик бурчтукту сызуу белгиси, эгер анын ордунда BF турса, анда боёлгон тик бурчтук сызылат.
Мисалы: SCREEN 7
LINE(110,70)-(190,120),6,B:LINE(0,0)-(320,200),3,BF

Айлананы CIRCLE(X,Y),<радиус>,<түс> операторунун жардамы менен сызабыз. Мында (X,Y) – борбордук чекиттин координаталары, <радиус> – айлананын радиусу. Мисалы:
SCREEN 7:CIRCLE(250,300),50,3

Эллипсти сызууда CIRCLE(X,Y),<радиус>,<түс>,,,<K> оператору колдонулат. Мында <K> - кысуу коэффициенти, K>1 болсо, анда эллипс Y огуна, K<1 болсо, анда эллипс X огуна карата кысылат, K=1 болсо, анда эллипс айланага барабар болот. Мисалы:
SCREEN 7
CIRCLE(320,200),40,4,,,1.5:CIRCLE(320,200),40,4,,,0.5
CIRCLE(320,200),40,4,,,1

CIRCLE(X,Y),<радиус>,<түс>,<бурч1>,<бурч2>,<K> оператору менен жааны сызсак болот. Мында <бурч1> – жаанын радиан менен берилген баштапкы бурчу, <бурч2> – жаанын радиан менен берилген акыркы бурчу, эгерде K нын мааниси берилбесе же K=1 болсо, анда айлананын, башка учурларда эллипстин жаасы сызылат.

Градустарды радианга которуу үчүн, градустун сандык маанисин PI/180 ге көбөйтүү керек.

PAINT(x,y),<түс>,<чектин түсү> оператору графикалык аймакты көрсөтүлгөн түстө же үлгүдө(толтурат) боёйт. Мында (x,y) – боёлуучу аймактын ичинде жаткан чекиттин координаты <түс> – боёчу түстүн атрибуту, <чектин түсү> – боёлуучу областын чегин көрсөтүүчү түс.
Мисалы: SCREEN 1
CIRCLE(106,100),75,1:LINE(138,35)-(288,165),1,B
PAINT(160,100),2,1

ТАТААЛ ГРАФИКАЛЫК ФИГУРАЛАР

Бейсикте сүрөттөлүштөрдү чыгарууда экрандын каалаган бөлүгүнөн тик бурчтуу аймак бөлүп алып иштөөгө болот. Ал үчүн VIEW,WINDOW операторлору колдонулат. VIEW SCREEN (X1,Y1)-(X2,Y2),C1,C2 оператору экранга түшө турган орунду жана анын көлөмүн аныктайт. Мында SCREEN – координаттар көрүү аймагына эмес, экранга сальштырмалуу берилерин көрсөтөт, (X1,Y1),(X2,Y2) – график түшө турган тик бурчтуктун диогоналдык чокуларынын координаталары, C1 – график түшө турган тик бурчтуктун ичинин түсү, C2 – график түшө турган тик бурчтуктун чегинин түсү; Мисалы:

```
SCREEN 1: VIEW(10,10)-(300,180),,1
LOCATE 4,4:PRINT "Графика көрүнө турган эң чоң аймак";
VIEW SCREEN (60,60)-(250,130),,1
LOCATE 11,11:PRINT "Графика көрүнө турган"
LOCATE 12,11:PRINT "эң кичине аймак";
```

WINDOW SCREEN(X1,Y1)-(X2,Y2) оператору график түшө турган тик бурчтуктун көлөмүн аныктайт. Тактап айтканда экранда аныкталган координаталар системасынан башка тик бурчтуу логикалык областын ичинен логикалык координаталар системасын аныктайт. Мында SCREEN – логикалык областын ичинлде Y тин маанисин жогортон төмөнкү мааниге өзгөртөт; (X1,Y1),(X2,Y2)- график түшө турган логикалык областын башкы диогоналынын чокуларынын координаталары; Мисалы:

```
SCREEN 7:VIEW(50,50)-(250,150),,7
WINDOW(-6.28,-1)-(6.28,1)
LINE(-6.28,0)-(6.28,0),3:LINE(0,-1)-(0,1),3
FOR X=-6.28 TO 6.28 STEP .1
PSET(X,SIN(X)),5:NEXT X:LOCATE 5,10:PRINT "sinx"
LOCATE 12,33:PRINT "X":LOCATE 6,20:PRINT "Y"
```

Бул программанын иштеши менен негизги экрандын координаталар системасынан сол жогорку чокусу (50,50), оң төмөнкү (250,150) чекиттеринин координаталары менен тик бурчтукту бөлүп алуу жана (-6.28,-1), (6,28,1) чокулары менен чектелген тик бурчтуу логикалык аймакта(координаталар системасында) sinx тин графиги чыгарылат. LINE оператору менен координаталык сызыктар сызылып, LOCATE оператору менен координаттык октор жана функциянын графиги боюнча түшүндүрмө берилет. Бул операторлор менен функциялардын графиктерин чыгаруу программаларын түзүү ыңгайлуу.

DRAW <командалар сабы$>обьект тартуу оператору б.а. сан менен көрсөтүлгөн сүрөттөлүштү чыгарат. Бул оператор менен чыга-

рылуучу кесиндилердин жардамында татаал сүрөттөлүштү жасоого болот. Мында: <командалар сабы$> – саптык туюнтма төмөндө келтирилген командалардын бир же бир нече удаалаштыгын аткарат.

Курсорду жылдыруу жана сызыкты сүрөттөө командалары:

Dn% – курсорду n% бирдикке төмөн жылдырат

Un% – курсорду n% бирдикке жогору карай жылдырат

Ln% – курсорду n% бирдикке солду карай жылдырат.

Rn% – курсорду n% бирдикке оңду карай жылдырат.

En% – курсорду n% бирдикке оңго жогору карай жылдырат

Fn% – курсорду n% бирдикке оңго төмөн карай жылдырат

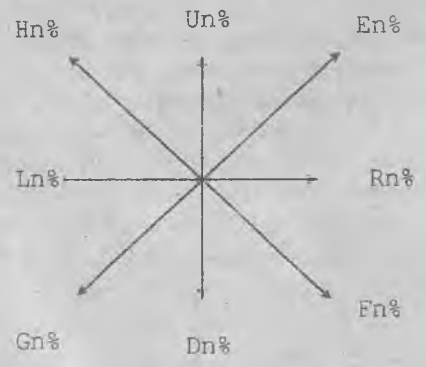
Gn% – курсорду n% бирдикке солго төмөн карай жылдырат

Hn% – курсорду n% бирдикке солго жогору карай жылдырат

B – курсорду из калтырбай которуу.

N – курсорду баштапкы абалына из калтыруу менен которуу.

M<{+же->X%,Y% – курсорду (X%,Y%) чекитине которот. Эгер координаталардын сандык маанилеринин астында "+" же "–" белгиси болсо, учурдагы чекитке салыштырмалуу жылдырат.



Түстөрдү берүү, буруу жана масштаб командалары:

An% – объектти n%*90 градуска бурат(n%=0,1,2 же 3)

Cn% – тартуу түсүн коет. n% – түстүн атрибуту

Pn1%,n2% – объекттин чегин же толтуруу түстөрүн коюу, n1% – толтуруу түсүнүн атрибуту, n2% – чектин түсүнүн атрибуту.

Sn% – курсордун которулуу узундугунун бирдигин орнотуу менен сүрөттөлүштүн масштабын аныктайт. Кадимки учурунда n%=4.

TAn% – n% градус бурчка бурат, мында n% –360 тан 360 ка чейинки маани алат.

Жогорудагы командалардын сан аргументтери өзгөрмө менен алмашылышы мүмкүн. Ал үчүн командадан кийин "=" белгиси коюлат. Эгер сызыкты сүрөттөө жана курсорду которуу командаларында n% ти калтырып койсок, анда курсор 1 бирдикке жылат.

1-Мисал: SCREEN 9:PSET (250,150),2
DRAW "U50G50R100H50BM250,150D50E50G50H50"
2-Мисал: COLOR 1,2:SCREEN 2
DRAW "BM128,96NS16R10U10L10D10S4"
3-Мисал: SCREEN 1:Triangle$ = "F60L120E60"
DRAW "C2X"+VARPTR$(Triangle$):DRAW"BD30P1,2C3M-30,-30"
4-Мисал: COLOR 1,15:SCREEN 7
DRAW "BM128,96":LINE STEP(0,0)-(60,60),5,B
5-Мисал: SCREEN 8
DRAW "BM30,50G3R50D40L50U40BM+55,+4H30G30BM+25,+16"
DRAW "G2R10D20L10U20BD6R10BL5D14"

GET STEP(X1,Y1)-STEP(X2,Y2),массивдин аты(индекс%) оператору – берилген сүрөттөлүшкө карата экрандагы информациялардын бардык чекиттерин окуп, сан маанисине сактоо оператору. Экрандын графикалык образын сактайт, файлды буферге окуйт. Мында: STEP – координаталар курсордун учурдагы графикалык абалына салыштырмалуу берилишин көрсөтөт. (X1,Y1)- сакталган образдын жогорку сол бурчунун координаталары, (X2,Y2) - сакталган образдын төмөнкү оң бурчунун координаталары. <массивдин аты> - образ сакталган массив, <индекс%>- образдын сакталуусун баштоочу массивдин индекси

PUT STEP(x1,y1), массивдин аты(индекс%),<кызм.сөз> оператору GET оператору сактаган образды экранга чыгарат. Мында: (x1,y1)- GET операторунда сакталган образдын жогорку сол бурчунун координатасы, <кызм.сөз> - образды чагылдырууну көрсөтүүчү төмөндөгү кызматчы сөздөрдүн бири:

Кызматчы сөз	Аракет
AND	Сакталган образды учурдагы образ менен кошот
OR	Сакталган образды учурдагы образга коёт(куюлушат).
PSET	Учурдагы образды өчүрүү менен сакталган образды тартат.
PRESET	Учурдагы образды өчүрүү менен сакталган образды реверсивдүү түстө тартат.
XOR	Мультипликациянын фондук эффектерин сактоо менен сакталган образды тартат же мурда тартылган образды өчүрөт.

GET оператору PUT оператору менен бирге жазылып иштөөсү менен динамикалык графика түзүүгө болот. Тик бурчтуу фрагменттер кыймылын башкарат. Кыймыл GET PUT операторлорунун жардамында "кадрлардын" удаалаш алмашышы менен жүргүзүлөт.

1-Мисал:
```
SCREEN 7:DIM Box%(1 TO 200)
x1%=0:x2%=10:y1%=0:y2%=10
LINE(x1%,y1%)-(x2%,y2%),2,BF
GET(x1%,y1%)-(x2%,y2%),Box%
DO: PUT (x1%,y1%),Box%,XOR
x1%=RND*300:y1%=RND*180
PUT(x1%,y1%),Box%
LOOP WHILE INKEY$=""
```

Тапшырма. Жогоруда келтирилген мисалдагы XOR кызматчы сөзүнүн ордуна таблицада келтирилген кызматчы сөздөрдү алмаштырып, программаны дагы бир жолу иштетип көргүлө.

2-Мисал:
```
REM кыймылдагы топ
SCREEN 1:CIRCLE(50,50),15,1
PAINT(50,50),1:DIM A(800):LOCATE 12,18:PRINT "ТОП"
GET(30,30)-(70,70),A:FOR I=30 TO 260 STEP .11
PUT(I,30),A:PUT(I+1,30),A,AND
NEXT I
```

3-Мисал:
```
REM кыймылдагы машина
COLOR 1,15:SCREEN 1:LINE(30,50)-(130,100),1,B
LINE(130,65)-(160,100),1,B:LOCATE 10,7
PRINT "НАН-ХЛЕБ":CIRCLE(50,100),15,2
CIRCLE(50,100),8,4:PAINT(50,100),4
CIRCLE(100,100),15,2:CIRCLE(100,100),8,4
PAINT(100,100),4:CIRCLE(145,100),13,2
CIRCLE(145,100),8,4:PAINT(145,100),4
LINE(135,75)-(150,95),1,BF:DIM A(1000)
GET (30,20)-(160,120),A
FOR i=30 TO 150 STEP .1:PUT(i,20),A
PUT(i+.1,20),A:NEXT i
END
```

IV БӨЛҮМ

ПРАКТИКАЛЫК ЖАНА ЛАБОРАТОРИЯЛЫК ИШТЕР

1. МАСЕЛЕЛЕР

№1 МАСЕЛЕЛЕР

Амалдарды аткаргыла.

1.	$11101_2+1010_2=$	4.	$1101110_2 : 1010_2=$	7.	$27_{10}+11011_2=X_{10}$
2.	$10111_2-1011_2=$	5.	$123_{10}=X_2=$	8	$11101_2+19_{10}=X_2$
3.	$1010_2*1011_2=$	6.	$101101_2=X_{10}$	9.	$1101_2+31_{10}-1010_2=$

№2 МАСЕЛЕЛЕР

Арифметикалык туюнтмаларды Паскаль(Бейсик) тилине которуп жазгыла.

1	$a=\dfrac{\sqrt{\lvert x-1\rvert}-\sqrt{\lvert y\rvert}}{1+\dfrac{x^2}{2}+\dfrac{y^2}{4}}$	6	$l=y+\dfrac{x}{y^2+\left\lvert\dfrac{x^2}{y+\dfrac{x^3}{3}}\right\rvert}$	9	$m=\left(y-\sqrt{\lvert x\rvert}\right)\cdot\left(x-\dfrac{y}{z^2+\dfrac{x}{4}}\right)$
2	$b=\dfrac{\lvert x\rvert-\lvert y\rvert}{1+\lvert xy\rvert}$	7	$z=(x+1)^2+\dfrac{1}{\sqrt{x+y}}$	10	$t=\left(2x^3y-1\right)^2+\dfrac{1}{\lvert x-y\rvert}$
3	$n=1+\dfrac{z^2}{3+\dfrac{z^3}{5}}$	8	$t=(1+y)\cdot\dfrac{x+\dfrac{y}{x^2+4}}{x^2+\dfrac{1}{y^2+4}}$	11	$x=\dfrac{y^{z^{\frac{a+b}{c}}}+\sqrt{y+10}}{c+d}$
4	$a=-0{,}8b^2+7{,}4b+(5{,}6b-0{,}2b^2)$				
5	$k=1+\lvert y-x\rvert+\dfrac{(y-x)^2}{2}+\dfrac{\lvert y-x\rvert^3}{3}$				

№3 МАСЕЛЕЛЕР

1.	a жана b деген чыныгы сандары берилди. Аладын суммасын, айырмасын жана көбөйтүндүсүн эсептөөчү алгоритмин жана программасын түз.
2.	x жана y деген чыныгы сандары берилди. $k = \frac{\lvert x\rvert - \lvert y\rvert}{1 + \lvert xy\rvert}$ туюнтмасын эсептөөчү алгоритмди жана программасын түз.
3.	Кубдун кыры белгилүү болсо, анын көлөмүн жана каптал бетинин аянтын эсептөөчү алгоритмди жана программасын түз.
4.	Эки чыныгы оң сандар берилди. Алардын арифметикалык жана геометриялык орто санын эсептөөчү алгоритмди жана программасын түз.
5.	Тик бурчтуктун узундугу, туурасы белгилүү болсо, аянтын, периметрин эсептөөчү алгоритмди жана программаны түз.
6.	Радиус белгилүү болсо, айлананын узундугун эсептөөчү алгоритмди жана программасын түз.
7.	h бийиктиктен ыргытылган таштын жерге түшүү убактысын аныктоочу алгоритмди жана программасын түз.
8.	Автомобилдин ылдамдыгы белгилүү болсо, 6 саат убакыт ичинде канча жол басып өтөөрүн аныктоочу алгоритмди жана программасын түз.
9.	x=1,8 z=-3,2 y= 18,2 $\gamma = \left\lvert x - \frac{y}{x}\right\rvert$ $\psi = \frac{(y-x)*(y-z/(y-x))}{1+(y-x)}$
10.	h=2,5 r=4,6 π=3,14 $r = \sqrt{2*r*h-h^2}$ $V = \frac{2\pi r^2 h}{3}$

№4 МАСЕЛЕЛЕР

1.	x жана y деген чыныгы сандары берилген. Бул эки сандын чоңун аныктоочу алгоритмди жана программасын түз.
2.	x, y, z чыныгы сандары берилген. Алардын: а) max(x, y, z) б) min(x+y+z, x*y*z) эсептөөчү алгоритмди жана програм-маны түз.
3.	x жана y чыныгы сандары берилген. $z = \begin{cases} x - y & x > y \\ y - x + 1 & x \le y \end{cases}$ туюнтмасын эсептөөчү алгоритмди жана программаны түз.
4.	Эки чыныгы сандары берилген. Эгерде биринчиси экинчисинен чоң болсо, биринчи сандын өзүн чыгаруучу, антпесе экөөнүн суммасын чыгаруучу алгоритмди жана програм-маны

	түз.
5.	Эки чыныгы сандары берилген. Эгерде биринчиси экинчисинен чоң же барабар болсо, анда биринчи сандын нөлгө айландыруучу, антпесе экинчи сандын өзүн чыгаруучу алгоритмди жана программаны түз.
6.	Үч чыныгы сан берилген. Үч сандын ичинен [1, 3] интервалында жаткан санды чыгаруучу алгоритмди жана программаны түз.
7.	x жана y чыныгы сандары берилген $(x \neq y)$. Кичинесинин ордун экөөнүн суммасы менен, ал эми чоңун экөөнү эки эселенген көбөйтүндүсү менен алмаштыруучу алгоритмди жана программаны түз.
8.	Үч чыныгы сан берилген. Оң сандарды квадратка көтөрүүнүн алгоритмин жана программасын түз.
9.	$k = \begin{cases} x+y & x \geq y \\ x-y & x < y \end{cases}$ туюнтманы эсептөөчү алгоритмди жана программаны түз.
10.	$\begin{cases} x & x \geq 1 \\ x+1 & -1 \leq x < 1 \\ x-1 & x \leq -1 \end{cases}$ туюнтманы эсептей турган алгоритмди жана программаны жаз.

№5 МАСЕЛЕЛЕР

Төмөнкү мисалдарды циклды уюштуруунун үч формасын колдонуп алгоритмин жана программасын түз.

1.	1 ден 10 го чейинки натуралдык сандардын суммасын аныктоо.
2.	Бир орундуу жуп сандардын көбөйтүндүсүн эсептөө.
3.	2 ден 50 гө чейинки жуп сандардын суммасын жана көбөйтүндүсүн аныктоо.
4.	1 ден 100 гө чейинки 3 кө бөлүнүүчү сандардын суммасын эсептөө.
5.	$k = \begin{cases} x^2 & x \geq 0 \quad x \in [-10, 10] \\ x^3 & x < 0 \quad \Delta x = 0{,}1 \end{cases}$
6.	$y = 2ax^2 - 1 \quad x \in [1, 2] \quad \Delta x = 0{,}1$
7.	$w = \begin{cases} at^2 + b\sin t + 1 & t \leq 0{,}1 \quad a = 2{,}5;\ b = 0{,}4 \\ at^2 + b\cos t + 1 & t > 0{,}1 \quad t \in [-1, 1];\ \Delta t = 0{,}2 \end{cases}$

8.	$v=\begin{cases} ax^2+bx+c & x<1,2 \quad a=2,8;\ b=-0,3;\ c=4 \\ \frac{a}{x}+\sqrt{x^2+1} & x=1,2 \quad x\in[1,2] \\ \frac{a+bx}{\sqrt{x^2+1}} & x>1,2 \quad \Delta x=0,05 \end{cases}$
9.	$y=\begin{cases} \sin x \lg x & x>3,5 \quad x\in[2,5] \\ \cos^2 x & x\le 3,5 \quad \Delta x=0,25 \end{cases}$
10	$v=\begin{cases} 1,5\cos^2 x & x<1 \quad a=2,3 \\ 1,8*a*x & x=1 \quad x\in[0,2;2,8] \\ (x-2)^2+6 & 1<x<2 \quad \Delta x=0,2 \\ 3tgx & x\ge 2 \end{cases}$

№6 МАСЕЛЕЛЕР

1.	A(N) массивдин элементтеринин суммасын жана санын эсептөөчү программаны түз.
2.	X(N) массивдин оң элементтеринин ордуна 1 деген санды, терс элементтеринин ордуна 0 деген санды басып чыгаруучу программаны түз.
3.	R(N) массивинин элементтеринин арифметикалык орто санын эсептөөчү программаны түз.
4.	Y(N) массивинин ар бир элементин эки эселенткен программаны түз.
5.	T(N) массивинин элементтерин өсүү тартибинде иреттөөнүн программасын түз.
6.	B(N) массивинин элементтеринин 3 кө бөлүнө турган элементтерин чыгаруучу программасын түз.
7.	U(N) массивинин эң чоң жана эң кичине элементерин таап, алардын ордун алмаштыргыла.
8.	H(N) массивинин адегенде оң элементтерин, андан кийин терс элементтерин A(N) массивине жайгаштыргыла.
9.	Q(N) массивинин эң кичине элементтин таап, анын катар номерин аныктагыла.
10.	P(N) массивинин бешке эселүү болгон элементтеринин эң чоңун аныктагыла.

№7 МАСЕЛЕЛЕР

1.	X(N,M) массивинин элементтерин сап(мамыча) боюнча суммалоо программасын түз.
2.	Z(N,M) массивинин негизги диогналынын элементтерин суммалоочу программаны түз.
3.	Y(N,M) массивинин негизги диогналынын жогору жагындагы элементтеринин көбөйтүндусүн чыгаруучу программасын түз.
4.	A(N,M) массивинин негизги диогналындагы элементтерди 0 го айландыруучу программаны түз.
5.	Y(N,M) массивинин тескери диогналынын төмөн жагындагы элементтеринин суммасын тап.
6.	X(N,M) массивин Y(N,M) массивине көбөйтүүчү программаны түз.
7.	X(N,M)массивинин ар бир сабынын элемент.еринин суммасын жана санын эсептегиле
8.	X(N,M) массивинин ар бир мамычасыдагы элементтерди өсүү тартибинде жайгаштыргыла.
9.	X(N,M) массивинин 3 кө эселүү болгон элементтерин таап, алардын эң чоңун чыгар.
10	X(N,M) массивинин негизги диогналынын жогору жагындагы элементтердин арифметикалык орто санын эсепте.

№8 МАСЕЛЕЛЕР

1.	Сап пробелдер менен бөлүнгөн. p-эсептөө системасындагы цифралардын тобунан түзүлгөн. Бул сандарды ондук эсептөө системасына караган символдордун сабын түзгүлө. Сандардын ортосун пробел менен бөлүп чыгаргыла. Саптын 1-сөзү эсептөө системасынын ($2 \le p \le 16$) негизи болуп саналат. Көрсөтүлгөн сап: 4 3 5 6 - 2 3 0
2.	Ар бир тилде оңдон солго, солдон оңго окулган бирдей сөздөр бар: казак, кудук, күлүк. Өзгөрмөлүү узундуктагы сөздөрдөн турган бир өлчөмдүү палиндромсөздөрдүн санын эсептегиле. Ошондой эле эң узун палиндромсөздү тапкыла.
3.	Бөлчөктү кыскартылбаган түргө алып келүүчү процедураны аныктагыла. Процедура 4 параметрге ээ болушу керек: берилген бөлчөктүн алымы жана бөлүмү, жыйынтыгындагы бөлчөктүн алымы жана бөлүмү.
4.	10-тартиптеги эки бүтүн сандуу квадраттык матрица берилген. Кайсынысынын эң кичине изи (из-башкы диогоналдык элементтеринин суммасы) болсо, анда анын элементтеринин суммасын эсептегиле. Эгер матрица-лардын издери барабар болсо, тиешелүү маалыматты чыгаргыла.

2. ЛАБОРАТОРИЯЛЫК ИШТЕР

№1 ЛАБОРАТОРИЯЛЫК ИШ

Экилик эсептөө системасында арифметикалык амалдарды аткаруу.

Иштин максаты: Экилик эсептөө системасында арифметикалык амалдарды аткарууну үйрөнүү жана машыгуу.

Түшүндүрмө

Экилик эсептөө системасында да арифметикалык амалдарды аткара алабыз. Ал үчүн төмөндөгү таблицаны колдонобуз.

Кошуу	Кемитүү	Көбөйтүү
0+0=0	0-0=0	0*0=0
0+1=1	1-0=1	0*1=0
1+0=1	0-1=1	1*0=0
1+1=10	1-1=0	1*1=1

Мисалы: кошуу

$$\begin{array}{rcccc} & 1 & 0 & 1 & 1_2 \\ + & & 1 & 1 & 1_2 \\ \hline 1 & 0 & 0 & 1 & 0_2 \end{array}$$

кемитүү

$$\begin{array}{rccccc} & 1 & 0 & 0 & 1 & 0_2 \\ - & & & 1 & 1 & 1_2 \\ \hline & & 1 & 0 & 1 & 1_2 \end{array}$$

көбөйтүү

$$\begin{array}{rccc} & 1 & 0 & 1_2 \\ * & & 1 & 1_2 \\ \hline & 1 & 0 & 1 \\ 1 & 0 & 1 & \\ \hline 1 & 1 & 1 & 1_2 \end{array}$$

бөлүү

$$\begin{array}{rcccc|cc} 1 & 1 & 1 & 1_2 & 1 & 0 & 1_2 \\ \hline - & 1 & 0 & 1 & & 1 & 1 \\ \hline & 1 & 0 & 1 & & & {\scriptstyle 2} \\ - & 1 & 0 & 1 & & & \\ \hline & & & 0 & & & \end{array}$$

Төмөнкү тапшырмаларды аткаргыла:

1.	$1101_2+101_2=$	5.	$110111_2-1010_2=$	9.	$1011_2+11001_2=$
2.	$1101_2*101_2=$	6.	$101101_2:101_2=$	10.	$1011_2+1101_2=$
3.	$11011_2+1110_2=$	7.	$10101_2-1101_2=$	11.	$101_2*10111_2=$
4.	$101_2*1101_2=$	8.	$101101_2:11011_2=$	12.	$1101_2:11_2=$

№2 ЛАБОРАТОРИЯЛЫК ИШ

Ондук эсептөө системасынан экилик эсептөө системасына жана экилик эсептөө системасынан ондук эсептөө системасына өтүү.

Иштин_максаты: Ондук эсептөө системасындагы сандарды экилик эсептөө системасына жана экилик эсептөө системасындагы сандарды ондук эсептөө системасына айландыруу жолдорун үйрөтүү, ошондой эле мисалдарды чыгара билүүгө жетишүү.

Түшүндүрмө

Биз "Эсептөө системасы" темасын өткөнүбүздө, анын түрлөрү жөнүндө кеп кылганбыз. Мына, ошол берилген ондук эсептөө системасындагы санды экилик эсептөө системасына өткөрө алабыз. Ал үчүн берилген ондук эсептөө системасындагы санды 2 ге улам бөлүп, тийинди бирге барабар болгнчо бул процессти улантабыз. Жыйынтыгын акыркы тийиндиден баштап, калдыктарын төмөндөн жогору көздөй жазабыз.

Мисалы: $35_{10}=X_2$

35	2				
- 34	17	2			
1	16	8	2		
	1	8	4	2	
		0	4	2	2
			0	2	1
				0	

Демек, $35_{10}=100011_2$

Мына ушул экилик эсептөө системасындагы санды кайра ондук эсептөө системасына өткөрүү жолдорун карайлы.

Мисалы: $100011_2=X_{10}$

1-жолу: $100011_2=1\cdot2^5+0\cdot2^4+0\cdot2^3+0\cdot2^2+1\cdot2^1+1\cdot2^0=$

$=32+0+0+0+0+2+1=35$

2-жолу:

	1	0	0	0	1	1_2		4		8		34	Демек, $100011_2=35$
*	2						+	0	*	2	+	1	
	2							4		16		35	
+	0						*	2	+	1			
	2							8		17			
*	2						+	0	*	2			
	4							8		34			

Төмөнкү тапшырмаларды аткаргыла:

1.	$92_{10}=X_2$	4.	$X_{10}=10101_2$	7.	$73_{10}=X_2$
2.	$1011_2=X_{10}$	5.	$X_{10}=110011_2$	8.	$11101_2=X_{10}$
3.	$28_{10}=X_2$	6..	$31_{10}=X_2$	9.	$10111_2=X_{10}$

№3 ЛАБОРАТОРИЯЛЫК ИШ

Сызыктуу алгоритм

Иштин максаты: Сызыктуу структурадагы мисал-маселелердин алгоритмин жана программасын түзүү жана практикалык иш аркылуу машыгуу менен өз алдынча иштөөгө жетишүү.

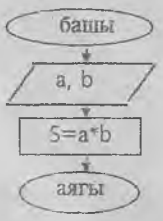
Түшүндүрмө

Мисалы: Тик бурчтуктун аянтын табуу.

1. Сөз түрүндө:
   1) тик бурчтуктун жактарын киргиз: a, b
   2) a ны b га көбөйт, жыйынтыкты S деп белгиле.
   3) жыйынтык S
2. Алгоритм тилинде: алг, башы, аягы деген кызматчы сөздөр колдонулат.

```
Алг тик бурчтуктун аянтын табуу
башы
    тик бурчтуктун жактарын киргиз
     a жана b ны көбөйт,
    жыйынтыкты S деп белгиле.
Аягы
```

3. Блок-схема түрүндө берилгенде геометриялык фигуралардын жардамы аркылуу берилет.



Төмөнкү тапшырмаларды аткаргыла:

1.	Тик бурчтуктун периметрин табуу.	6.	Айлананын узундугун табуу.
2.	Чай кайнатуу.	7.	Телефон менен сүйлөшүү.
3.	Үй куруу.	8.	Тегеректин аянтын табуу.
4.	Бак отургузуу.	9.	Тамак жасоо.
5.	Мектепке келүү.	10.	Китеп окуу.

№4 ЛАБОРАТОРИЯЛЫК ИШ

Тармактуу алгоритм

Иштин максаты: Тармактуу структурадагы эсептөө процессинде алгоритмдин берилиш жолдорун пайдаланып, алгоритм жана программаны түзүүгө үйрөнүү.

Түшүндүрмө

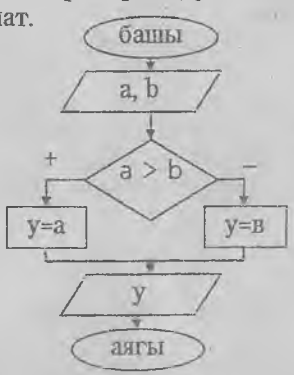
Мисалы: Эки сандын чоңун табуу

1. Сөз түрүндө:
   1) a жана b сандарын киргиз.
   2) a жана b сандарын салыштыр.
   3) эгерде a саны b дан чоң болсо, анда y=b деп белгиле.
   4) антпесе y=b деп белгиле.
   5) жыйынтыкты y деп ал.

2. Алгоритм тилинде тармактуу алгоритм эгер, анда, антп деген жардамы жазылат.

```
Алг  Эки сандын чоңун табуу
 башы
      эгер a>в
            анда  y = a
            антп  y = b
            жый   y
      бүт
 аягы
```

3. Блок-схема түрүндө берилишинде шарт ромбдун ичине аркылуу берилет.



Төмөнкү тапшырмаларды аткаргыла:

1.	Телевизор көрүү	6.	Компьютерде иштөө
2.	Экскурцияга баруу.	7.	Тамак жасоо
3.	Базарга баруу.	8.	Жолдон өтүүнүн алгоритми.
4.	“Ак терек, көк терек” оюунун ойноонун алгоритми.	9.	Таандык жак -сы(-ы) мүчөсүн улап жазуунун алгоритми.
5.	“Улак тартмай” оюунун ойноонун алгоритми.	10.	“Кыз куумай” оюунун ойноонун алгоритми

№5 ЛАБОРАТОРИЯЛЫК ИШ

Циклдуу алгоритм

Иштин максаты: Циклдуу структурадагы эсептөө процессинде алгоритмдин берилиш жолдорун пайдаланып, алгоритм жана программаны түзүүгө үйрөнүү.

Түшүндүрмө

Мисалы: Бир жуманын ичинде мектепке келүү.

1. Сөз түрүндө:
   1) эртең менен тур, кийимиңди кий, тамактан, сыртка чыгып, аялдамага бар, автобуска түш, же жөө мектепке бар, мектепке кел, сабактарга кирип, окуу.
   2) качан жекшемби болгончо, жогорудагы аракетти аткар.
2. Алгоритм тилинде азыр, цб. ца кызматчы сөздөр колдонулат.

Алг бир жумада мектепке келүү
башы
  азыр жекшемби эмес
  цб
  эртең менен тур, кийимдериңди кий, тамактан, сыртка чыгып, автобуска түшүп же жөө мектепке кел, сабакты окуу
  ца
аягы

3. Блок-схема түрүндө:

Төмөнкү тапшырмаларды аткаргыла:

1.	Бош чаканы кружка менен толтуруу.
2.	2 ден 20 га чейинки жуп сандардын суммасын табуу
3.	1 ден 10 го чейинки сандардын суммасын табуу.
4.	1 ден 100 гө чейинки 5 саны бөлүнүүчү сандарды табуунун алгоритмин түзүү.
5.	"Домино" оюнун ойноонун алгоритми.

№6 ЛАБОРАТОРИЯЛЫК ИШ

Паскаль жана Бейсик тилдеринде туюнтмалардын жазылышы.

Иштин максаты: Математикалык туюнтмаларды Паскаль жана Бейсик тилдеринде жазууга машыгуу жана калыптануу.

Түшүндүрмө

Паскаль жана Бейсик программалоо тилдеринде латын, орус тамгалары, араб цифралары, арифметикалык амалдар мисалдарды иштөөдө колдонулат, бирок математикалык мисалдарды иштөөдө амалдарды бүтүн сандарды, даража көрсөткүчтөрдү атайын программалоо тилинде жазуу зарыл.

Математикалык жазылышы	Паскаль тилинде жазылышы	Бейсик тилинде жазылышы
0,75	0.75	0.75
–	–	–
+	+	+
•	*	*
:	/	/
$\|x\|$	ABS(x)	ABS(x)
x^2	SQR(x)	x^2
$\sqrt{x}$	SQRT(x)	SQR(x)
x^n	–	x^n
cosx	COS(X)	COS(X)
sinx	SIN(X)	SIN(X)
tgx	SIN(X)/COS(X)	TAN(X)

Мисалы: $2x^2+\frac{1}{|x+2|}+\sqrt{3}-0,21$ туюнтмасын программалоо тилдеринде жазсак, анда төмөндөгүдөй болот:

Паскаль тилинде: 2*SQR(x)+1/ABS(X+2)+SQRT(3)-0.21

Бейсик тилинде: 2*x^2+1/abs(x+2)+sqr(3)-0.21

Төмөндөгү мисалдарды программалоо (Паскаль, Бейсик) тилдеринде жазгыла

1	$\frac{x^2+y^2}{1-\frac{x^2+y^2}{2}}$	2	$\frac{a+b-1,7}{c+\frac{d}{e+f+0,5}}$	3	$\frac{a+b}{c+d}-2,5$	4	$\left(x+\frac{x-1}{y+1}\right)y+\frac{a-x^2}{b+y}$
5	$1+x+\frac{x^2}{2}$	6	$\frac{1,2-9,8x}{1-y^2}$	7	$\frac{x^3-y^3}{y}+\frac{x}{x^3-y^3}$	8	$\frac{x^2-y^3}{y}+\frac{x}{x^3-y^3}$
9	$1+\|x\|+\|1+x\|$			10	$\sqrt{1+\sqrt{\|x\|}}$		

Төмөндөгү программалоо тилдеринде жазылган туюнтмаларды математикалык түргө алып келгиле

1.	1+sqr(abs((x+y)/2)
2.	sqrt(a+b)-sqrt(a-b)
3.	a*b/(c+d)-(c-d)/b*(a+b)
4.	a+b/(c+d)-(a+b)/c+d
5.	(b+sqrt(sqr(b)-4*a*c))/2*a

№7 ЛАБОРАТОРИЯЛЫК ИШ

Менчиктөө оператору

Иштин максаты: Паскаль жана Бейсик тилиндеги менчиктөө операторуна мисалдарды иштөө менен, алгоритм жана програм-маны түзүүгө үйрөнүү.

Түшүндүрмө

Паскаль тилиндеги менчиктөө операторунун жалпы жазылышы төмөндөгүдөй болот: <өзгөрмө>:=<арифметикалык туюнтма>

Бейсик тилинде менчиктөө оператору – Let. Бул операторду жазсак да болот, жазбасак да болот.

let <өзгөрмө> = <арифметикалык туюнтма>

Менчиктөө операторунда оң жактагы арифметикалык туюнтманын мааниси эсептелинип, сол жактагы өзгөрмөгө менчиктелет, ыйгарылат, башкача айтканда таандык кылынат.

Мисалы: $y = 2x^2 + |x|$

Паскаль тилинде: y:=2*SQR(X)+ABS(X);

Бейсик тилинде: LET y=2*X^2 + ABS(X)

Төмөнкү тапшырмаларды аткаргыла:

1.	a:b-SQR(2) ыйгаруу оператору болобу? Болсо, эмне үчүн?
2.	a*в+x:=0 ыйгаруу оператору болобу? Болсо, эмне үчүн?
3.	z өзгөрмөсүнө x жана y өзгөрмөлөрүнүн суммасынын жарымын ыйгарып жазгыла.
4.	k өзгөрмөсүнө a жана b өзгөрмөлөрүнүн суммасын, ал эми t өзгөрмөсүнө сумманы ыйгарып жазгыла.
5.	r өзгөрмөсүнө a өзгөрмөсүн эки эселентип ыйгарып жаз.
6.	p өзгөрмөсүнө өзүн 0,1 ге чоңойтуп ыйгарып жаз.
7.	x өзгөрмөсүнө a жана b өзгөрмөлөрүнүн ариф-к орто санын, y өзгөрм-о бул сандардын геом-к орто санын ыйгарып жазгыла.
8.	r өзгөрмөсүнө y жана z өзгөрмөлөрүнүн эки эселенген кобойтундусун t өзгөрмөсүнө 0 санын ыйгарып жаз.
9.	r өзгөрмөсүнө a жана b өзгөрмөлөрүнүн суммасынын квадратынан бирди кемиткенди менчиктеп жаз.
10	b өзгөрмөсүнө y менен 1 дин айырмасынын абсолюттук чоңдугун менчиктеп жаз.

№8 ЛАБОРАТОРИЯЛЫК ИШ

Паскаль тилинин (write, writeln) жана Бейсик тилинин (print) чыгаруу операторлору.

Иштин максаты: Паскаль жана Бейсик тилдерин пайдаланып, математикалык туюнтмаларды программалоо тилдерине которуп жазууга үйрөнүү жана калыптануу.

Түшүндүрмө

Паскаль тилиндеги чыгаруу опреатору – бул write, writeln. Бул экөө тең чыгаруучу оператору болуп эсептелет. write,writeln "жазуу" дегенди билдирет. write маалыматты бир сапка эле чыгарат, ал эми writeln оператору болсо, курсорду кийинки сапка түшүрүп, анан чыгарат. Бул эки операторду тең колдонсок болот. Анын жалпы жазылышы төмөндөгүдөй болот:

write (<өзгөрмөлөрдүн тизмеси>);

writeln(<өзгөрмөлөрдүн тизмеси>);

Бейсик тилиндеги чыгаруучу оператор болуп – print эсептелет. Жазылуу форматы: print <өзгөрмөлөрдүн тизмеси>

Мисалы: 1. "Менин атым Мерует" - деген сөздү чыгарууда

Паскаль тилинде: write ('Менин атым Мерует');

Бейсик тилинде: print "Менин атым Мерует"

деп жазабыз.

2. k=12 + 79 – 2 туюнтмасын жазып, чыгаруу үчүн

Паскаль тилинде: write('k=12+79-2');

Бейсик тилинде: print "k=12+79-2"

3. 12 + 79 – 2 туюнтмасынын жыйынтыгын чыгаруу үчүн

Паскаль тилинде: write('k=',12+79-2);

Бейсик тилинде: print "k=";12+79-2

десек, анда жыйынтыгында k=89 деген жооп чыгат.

Төмөнкү тапшырмаларды аткаргыла (Паскаль жана Бейсик тилдеринде):

1	"Мен Паскаль тилинде программа түздүм" – деген сөздү чыгар.
2	"Мен Бейсик тилинде программа түздүм" – деген сөздү чыгар.
3	679-71+41 туюнтманын өзүн чыгаруучу программа.
4	679-71+41 туюнтманын жыйынтыгын чыгаруучу программа түз.
5	x=0,2 болсо $k=x^2+2$ туюнтмасын эсептеп чыгаруучу алгоритм жана программасын түз.
6	Каалаган диалогдук программаны түз.

7	Өз атыңды, фамилияңды, туулган жылыңды киргизип, экранга чыгаруучу программа түз
8	"Биздин класс" - деген темада чакан аңгеме түзүп, аны экранга чыгар
9	Төмөнкү жазылыштардын кайсынысы туура: а) read(x,y); x=y+1; write(x,y); б) read(x,a,b); a:=b*x; write(a,b,x); в) read(x,y,z) d:=x+y+z write(x,y,z,d); г) read x,y; write x,y; д) read(x,y); y:=x+5; write('y=',y:8:2); е) write("КИТЕП"); ж) write('Биздин класс ынтымактуу');
10	Төмөнкү жазылыштардын кайсынысы туура: а) input x,y : print x,y б) input a,b,c print a,b,c в) input d,k,n : z=d+n-c : print d,n,c,z г) input(x,y,z); д) print 'Калем сап' е) input "x=";x,"y=";y ж) print "x==";x,"y=";y

№9 ЛАБОРАТОРИЯЛЫК ИШ

Сызыктуу программалоо

Иштин максаты: Паскаль жана Бейсик тилдерине сызыктуу программа түзүү менен бирдикте, билгичтиктерин калыптандыруу.

Түшүндүрмө

Мисалы: Айлананын узундугун эсептөөчү программаны түзүү.

Паскаль тилинде:

```
Program ailana;
const p=3.14;
var r,c:real;
begin
writeln('r ди киргиз');
read(r);
c:=2*p*r;
write('c=',c:6:2);
end.
```

Бейсик тилинде:

```
REM ailana
P=3.14
PRINT "r ди киргиз"
INPUT r
C=2*P*R
PRINT "C=";C
END
```

Төмөнкү тапшырмаларды аткаргыла:

1.	Тик бурчтуктун аянтын жана периметрин аныктоочу программаны түз.
2.	$a=6$ болсо $k=\lvert 2a^2-1\rvert+27$ туюнтмасынын маанисин эсептөөчү программаны түз.
3.	$w=\frac{2a^2}{7}-15x^3-23 \quad a=2{,}1$ туюнтмасынын маанисин эсептөөчү программаны түз.
4.	Герондун формуласын колдонуп, үч бурчтуктун аянтын эсепте.
5.	Радиусу 16 см ге барабар болсо, анда айлананын узундугун эсепте.
6.	Тик бурчтуктун аянты 120 $см^2$, ал эми узундугу 20 см ге барабар болсо, Туурасын эсепте.
7.	$v=\sqrt{(x-1)+x^2}+\frac{1}{3}x$ туюнтмасынын маанисин эсептөөчү программаны түз.
8.	$t=\frac{x-y}{x+y}-2xy$ туюнтмасынын маанисин эсептөөчү программаны түз.

9.	$t=\frac{1}{x+\frac{x}{1+x}}+2(x-y)$ туюнтмасынын маанисин эсептоочу программаны түз.
10	a, b, c, d, e, f, g каалагандай натуралдык сандар берилген. Бул сандардын арифметикалык орто сапын эсептөөчү программаны түз.

№10 ЛАБОРАТОРИЯЛЫК ИШ

Экранды жасалгалоо

Иштин_максаты: Паскаль тилинде программа түзүү менен бирдикте экранды жасалгалоону үйрөнүү, практикада колдонуу.

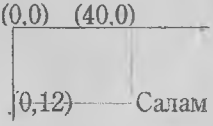
Түшүндүрмө

Паскаль тилинде программа түзүү менен экранды ар түрдүү жасалгалоону жүргүзө алабыз.

1. Экранды жасалгалөодо CRT модулунун колдонобуз. Ал үчүн программанын аталыш аймагында Uses crt; ны жазуу жетиштүү
2. textbackground(n) – бул монитордун экранынын түсүн өзгөртөт. ($0 \le n \le 15$)
3. textcolor(n) - бул текстин түсүн өзгөртөт. ($0 \le n \le 15$)
4. gotoxy(x,y) - бул чыгарылуучу маалыматты монитордун каалагандай жерине чыгарууга жардам берет. $0 \le x \le 80$, $0 \le y \le 25$.
5. clrscr – экранды тазалайт.

Мисалы: Көк фонго сары түс менен "Салам" деген сөздү мони-тордун экранынын ортосуна чыгаруучу программа төмөндөгүдөй болот:

```
program gh;
uses crt;
begin
textbackground(1);
textcolor(14);
clrscr; gotoxy(40,12);
write('Салам');
end.
```



Төмөнкү тапшырмаларды аткаргыла:

1.	Экрандын оң жак жогорку бурчуна жашыл фонго, кызыл түс менен өз атыңды чыгаруучу программаны түз.
2.	Экрандын сол жак жогорку бурчуна кызыл фонго, кара түс менен "Улуттук компьютердик гимназия" деген сөздү чыгаруучу программаны түз.
3.	"2001-жыл туризм жылы " ********** деген маалыматты монитордун ортосуна көк түс менен чыгаруучу программаны түз.
4.	"Силер XXI кылымдын кадрларысыңар!" деген сөздү монитордун жогорку сол бурчуна көк фонго сары түс менен чыгаруучу программаны түз.
5.	"МАНАС эпосу кыргыз элинин энциклопедиясы" деген сөздү монитордун төмөнкү оң бурчуна жашыл фонго кызыл түс менен чыгаруучу программаны түз.
6.	$y=-2x^2$ жана $y=2x^2$ функцияларынын маанисин жасалгалоо менен жыйынтыгын чыгаруучу программаны түз.
7.	"Менин үй-бүлөм"- деген темада чакан анкета жазып, аны жогоруда көрсөтүлгөн экранды жасалгалоо операторло-рунун жардамы менен экранга чыгар.
8.	"Кыргызстан – көп улуттуу өлкө"- деген сүйлөмдү экрандын ортосуна кызыл түстө сары фондо чыгаргыла.

№11 ЛАБОРАТОРИЯЛЫК ИШ

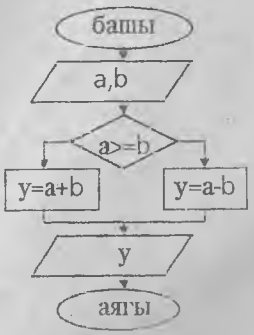
Шарттуу операторлор

Иштин максаты: Шарттуу операторлорго мисалдар иштөө менен бирдикте программа түзүүгө такшалуу.

Түшүндүрмө

Мисалы: $y=\begin{cases} a+b & a \geq b \\ a-b & a < b \end{cases}$ Алгоритм тилинде:

```
Алг
  башы a,b
    эгер a >= b
       анда y = a + b
       антп y= a – b
       жый y
    бүт
  аягы
```



Паскаль тилинде:	Бейсик тилинде:

program tuiuntma;
var a,b,y:real;
begin
write('a,b ны киргиз');
read(a,b);
if a>=b then y:=a+b else y:=a-b;
write('y=',y:6:4);end.
	REM tuiuntma
	INPUT "a,b ны киргиз";a,b
	IF a>b THEN y=a+b ELSE y=a-b
	PRINT y
	END

Төмөнкү тапшырмаларды аткаргыла:

1.	$z=\begin{cases} a^2+b^2x & x\le 0 \\ \sqrt{ax}+\dfrac{bx^2}{c} & x>0 \end{cases}$ $a=0{,}95;\ b=3{,}45$
2.	$k=\begin{cases} 5y+\dfrac{y^2-5y+7}{\lvert y+1\rvert} & y\ge 10 \\ y^2+7y-3 & y<10 \end{cases}$
3.	$t=\begin{cases} \dfrac{z-1}{5}+\sqrt{z} & z\ge 30 \\ \dfrac{\lvert z+7\rvert}{z}+z^2 & z<10 \end{cases}$
4.	Кандайдыр бир b саны берилген. Эгерде b саны оң сан болсо, анда аны квадратка көтөрүп, жыйынтыгына 5 ти кош, антпесе b санынын абсолюттук модулун эсепте
5.	d жана q сандары берилген. Эгерде бул сандардын суммасынын кубу, алардын квадраттарынын суммасынан чоң болсо, анда ал сандардын көбөйтүндүсүн, антпесе катышын тапкыла. $d\neq 0$, $q\neq 0$
6.	x саны берилген. $0\le x\le 1$ барабарсыздыгынын аткарылышын текшергиле. Эгер бул барабарсыздык аткарылса, анда 1 санын, антпесе 0 санын экранга чыгаргыла.
7.	x, y ,z сандары берилген. max(x+y+z,xyz)+5 туюнтмасынын маанисин эсептегиле.
8.	x, y ,z сандары берилген. min(x^2+y^2, y^2+z^2)–5 туюнтмасынын маанисин эсептегиле.
9.	x,y өзгормолөрү берилген. Координатасы(x,y) болгон чекит, борбору координаталар башталышында жаткан бирдик радиустагы айланада жатаарын аныктагыла.
10	Үч сан берилген. Бул сандардын арасынан терс болгондорун квадратка которуу программасын түз.

№12 ЛАБОРАТОРИЯЛЫК ИШ

Тармак ичинде тармак

Иштин максаты: Тармак ичинде тармакка мисалдар иштөө менен бирдикте программа түзүүгө такшалуу.

Түшүндүрмө

Турмуштагы аракеттер бир эле шарттан көз каранды эмес, бир нече шарттардан көз каранды болушу мүмкүн.

Мисалы: $y=\begin{cases} x & x\geq 2 \\ x+1 & 0\leq x<2 \\ x-1 & x\leq 0 \end{cases}$

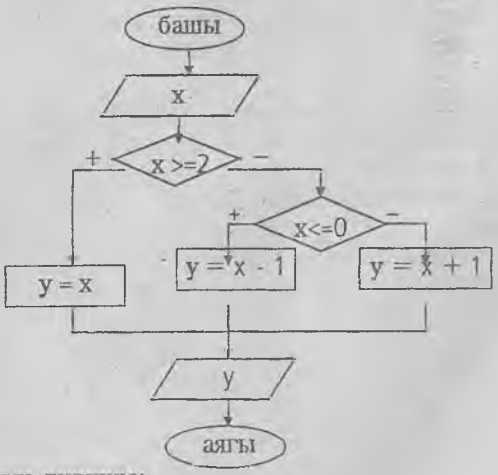
Жогорудагыдай мисалдар берилсе, биз аны чыгаруу үчүн эки жолу шарт текшеребиз. Анын жалпы жазылыш төмөндөгүдөй болот:

If <шарт 1> then A

ELSE If <шарт 2> then B ELSE C

A, B, C-аракеттер.

Блок-схемасы:



Паскаль тилинде:

```
program tuyuntma1;
var x,y:real;
 begin
  write('x тин маанисин киргиз'); read(x);
  if x>=2 then y:=x else if x<=0 then y:=x-1
  else y:=x+1;
  write ('y=',y:6:4);
end.
```

Бейсик тилинде:

```
REM
INPUT x
IF x>=2 THEN y=x ELSE IF x<=0 THEN y=x-1 ELSE y=x+1
PRINT y
END
```

Төмөнкү тапшырмаларды аткаргыла:

1.	a, b, c сандары берилген. a<b<c барабарсыздыгы аткарылса "туура", анпесе "туура эмес" деген маалыматты экранга чыгаргыла.
2.	x,y сандары берилген. Эгер экөө тең терс сан болсо, анда алардын ар бирин модулдары менен алмаштыр, эгер бирөө гана терс болсо, анда алардын маанисин 5 ке чоңойт, эгер экөө тең оң сан болсо, анда алардын суммасын эсепте.
3.	Эгер берилген x, y, z сандарынын суммасы бирден кичине болсо, анда x жана y тин кичинесин y менен ż тин жарым суммасы менен алмаштыр
4.	x, y, z оң сандары берилген. Жактары x, y, z болгон үч бурчтукту чийүүгө болобу? Жообун текст формасында чыгаргыла.
5.	Берилген төрт сандын эң кичине маанисин тапкыла.
6.	Берилген сандардын арасынан оң сандарды таап, алар жуп экендигин аныктагыла.
7.	Берилген сандардын арасынан3 кө так бөлүнгөн сандардын чоңун тапкыла. ?
8.	Эгерде x, y сандарынын суммасы z, t сандарынын суммасына барабар болсо, анда x менен t нын жана y менен z тин айырмаларын салыштыргыла.

№13 ЛАБОРАТОРИЯЛЫК ИШ

Goto шартсыз өтүү оператору.

Иштин_максаты: Шартсыз өтүү операторуна мисалдар иштөө менен бирдикте программа түзүүгө такшалуу.

Түшүндүрмө

Мисалы: ax = b түрүндөгү теңдемени чыгаруунун программасы

Паскаль тилинде:

```
program tendeme;
label 10, 20;
var a,b,x:real;
begin
write('a,b ны киргиз');
read (a,b);
if a=0 then goto 10 else x:=b/a;
write('x=',x:5:3);goto20;

10: write('теңдеменин тамыры жок');
20: end.
```

Бейсик тилинде:

```
REM
INPUT a,b
IF a=0 THEN GOTO 10 ELSE
x=b/a:GOTO 20
10 PRINT "чыгарылышы жок"
20 PRINT x
END
```

Төмөнкү тапшырмаларды аткаргыла.

1.	x, y, z сандарынын эң кичинесин тапкыла.
2.	Берилген үч сандардын бири-бири менен өз көбөйтүндүлөрүн салыштыргыла. Мисалы: a, b, c; a*b, b*c, a*c
3.	$z = 4x^2 + 5$ теңдемесинин бардык тамырларын тапкыла
4.	Төрт сан берилген. Эгерде алардын суммасыны ар бир сандын квадратынан чоң болсо, анда ал сандарды ирети менен жайгаштыр
5.	Ар кандай геометриялык фигуралар берилген. Алардын аянттарын салыштыргыла.
6.	Берилген сандардын көбөйтүндүсү алардын суммасынын квадратына барабар болсо, ал сандардын ар бирин кубга көтөр.
7.	Бир класста 15 окуучу бар. Алардын чейрек боюнча алган баалары боюнча канчоосу "2", "3", "4", "5" деген бааларды алганын аныкта.
8.	$y = \begin{cases} x+1 & x<10 \\ x^2+2x-1 & 10<x<20 \\ \lvert x-5\rvert & x>20 \end{cases}$ туюнтмасын эсептегиле

№14 ЛАБОРАТОРИЯЛЫК ИШ

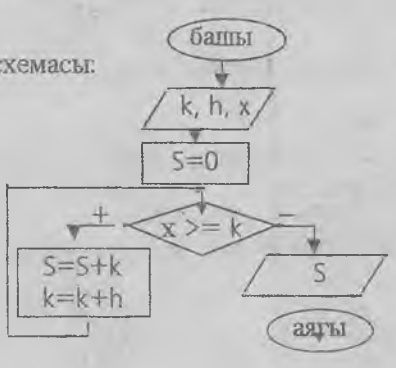
Азырынча цикли

Иштин максаты: Программалоо тилдерине мисалдар иштөө менен билимдерин такшалтуу.

Түшүндүрмө

Мисалы: 1 ден 20 га чейинки жуп сандардын суммасын табуу.

Блок-схемасы:



Паскаль тилинде:	Бейсик тилинде:

Program summa;
var s,k,x,h:real;
begin
write('k,x,h ты киргиз');
read (k,x,h);s:=0;
while x>=k do begin
s:=s+k; k:=k+h; end;
write('k=',k:6:4,'s=',s:5:3);
end.
	REM
	INPUT k,x,h
	s=0
	10 IF x>=k THEN GOTO 20
	s=s+k:k=k+h: GOTO 10
	20 PRINT s
	end

Төмөнкү тапшырмаларды аткаргыла:

1.	1 ден 50 ге чейинки так сандардын суммасын тапкыла.
2.	$z = \sqrt{y^2 + 5y - 4} + \lvert 4y^2 - 7y + 16 \rvert$ функциясынын маанисин [-2, 2] аралыгында 0,25 кадамы менен эсептеп, таблица формасында чыгар
3.	а оң саны берилген. а дан кичине болгон эң чоң $\frac{1}{2^n}$, $n \geq 0$ санын тапкыла
4.	1 ден 5 ке чейинки сандардын ар биринин факториалын эсептегиле. Мисалы: 1!, 2! , ж.б.

5.	Эгерде эки сандын суммасы N ден чоң болсо, анда ал сандардын квадраттарынын суммасын эсепте
6.	a жана b сандары берилген(a>1). b санынан кичине болгон a, a^2, a^3, . . . чексиз удаалаштыгынын мүчөлөрүн тапкыла.
7.	Анык сандардын удаалаштыгында терс жолукканга чейинки бардык сандардын суммасын эсептегиле.
8.	Анык сандардын удаалаштыгынан терс сан жолукканга чейин бардык санды квадраттык тамырдан чыгаргыла.

№15 ЛАБОРАТОРИЯЛЫК ИШ

Чейин цикли (repeat, until)

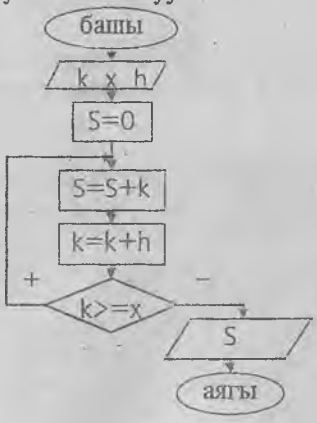
Иштин максаты: Программалоо тилдеринде чейин циклине мисалдар иштөө менен билимдерин такшалтуу.

Түшүндүрмө

Мисалы: 2 ден 20 га чейин жуп сандардын суммасын табуу

Паскаль тилинде

```
program sum;
var s,k,x,h:real;
begin
write ('k,x,h, киргиз');
read(k,x,h); s:=0;
repeat
s:=s+k; k:=k+h;
until k>=x ;
writeln ('s=',s:5:3);
end.
```



Бейсик тилинде:

```
REM
INPUT k,x,h:s=0
10 s=s+k:k=k+h
IF k>x THEN GOTO 10
PRINT s:END
```

Төмөнкү тапшырмаларды аткаргыла:

1.	n санынын бардык натуралдык бөлүүчүлөрүн экранга чыгар.
2.	n жана m сандарынын жалпы орток бөлүмүн тап.
3.	1 ден 100гө чейинки сандардан x үчүн көп мүчөнүн маанисин чыгар(x типар бир кийинки мааниси мурункусунан 1 ге чоң).
4.	10 дон 20 га чейинки натуралдык сандардын кубун экранга чыгар.
5.	$a_1,\ldots,a_n$ анык сандары берилген. a га бөлүнүүчү жана бөл

	бөлүнбөөчү сандардын санын жана суммасын тап.
6.	n натуралдык, $a_1,...a_n$ чейинки сандар берилген. $a_1,....a_n$ эң чоң жети мүчөнү 7 саны менен алмаштыр жана мындай мүчөлөрдүн санын эсепте.
7.	n натуралдык саны, a_1 , ..., a_n чейинки бүтүн сандары берилген. a1,..., a_n чейинки оң сандардын суммасын жана терс сандардын (эсебин)санын алгыла.
8.	n натуралдык саны, a, x_1 ,..., x_n бүтүн сандары берил-ген. x_1, ..., x_n арасында жок дегенде a га барабар болгон мүчө бар болсо, анда андан кийинки бар-дык мүчөлөрдүн суммасын, антпесе жообун a саны болгудай кылып эсептегиле.

№16 ЛАБОРАТОРИЯЛЫК ИШ

Параметрлүү циклды уюштуруу (For, Do командалары)

Иштин максаты: Программалоо тилдеринде циклды уюштуруу-нун атайын формаларына мисалдар иштөө менен билимдерин такшалтуу.

Түшүндүрмө

Паскаль тилинде параметрлүү циклды уюштурууда өзгөрмөнүн баштапкы жана акыркы маанилери менен иш алып барабыз жана өзгөрмөнүн баштапкы мааниси улам бирге гана чоңоёт, кадамды бирден чоң деп ала албайбыз. Параметрлүү циклдын жазылуу форматы: for i:=k to n do

мында k – өзгөрмөнүн баштапкы мааниси

n – өзгөрмөнүн акыркы мааниси

Ал эми Бейсик тилинде мындай проблемадан чыгуу үчүн STEP оператору колдонулат, б.а. өзгөрмөнүн маанисин бирге гана чоңойтпостон каалаган өсүндүгө чоңойто алабыз. Жазылуу форматы:

FOR i=k TO n STEP d

мында k – өзгөрмөнүн баштапкы мааниси, n – өзгөрмөнүн акыркы мааниси, d – өсүндү

Мисалы: 1 ден 10 го чейинки сандардын суммасы.

Паскаль тилинде:	Бейсик тилинде:

Program summa;
var s:real;k:integer;
begin s:=0;
for k:=1 to 10 do s:=s+k;
writeln('s=',s:5:3);
end.
	REM
	s=0:FOR k=1 TO 10
	s=s+k
	NEXT k
	PRINT "s=";s
	END

Төмөнкү тапшырмаларды аткаргыла:

1.	n натуралдык саны берилген. Төмөнкүнү эсептегиле: $\frac{1}{1}+\frac{1}{2}+\frac{1}{3}+...+\frac{1}{n}$
2.	n натуралдык саны, анык a_1, ..., a_n сандарынын удаалаштыгы берилген. a_1, ..., a_n дин орточо арифметикалык санын эсептегиле.
3.	Анык a_1, ..., a_n сандарынын удаалаштыгы берилген. Бул удаалаштыктын мүчөлөрүнүн арасынан 5 ке бөлүнгөндөрүнүн суммасын эсептегиле.
4.	n натуралдык саны, анык y_1, ..., y_n сандарынын удаалаштыгы берилген. Төмөнкүлөрдү эсептегиле: $\max(\lvert z_1\rvert,...,\lvert z_n\rvert)$, мында $z_i=\begin{cases} y_i & \lvert y_i\rvert \le 2 \\ 0{,}5 & \text{башка убакта} \end{cases}$
5.	n натуралдык саны, анык y_1, ..., y_n сандарынын удаалаштыгы берилген. Төмөнкүлөрдү эсептегиле: $z_1^2+...+z_n^2$, мында $z_i=\begin{cases} y_i & 0\le\lvert y_i\rvert\le 10 \\ 1 & \text{башка убакта} \end{cases}$
6.	y=sinx функциясынын маанилер таблицасын чыгаргыла. Таблица эки мамычага ээ, анын i-сабында тиешелүү түрдө x_i жана y_i жайгашкан. Мында x_i=0,1i, y_i=sinx_i.
7.	n дин каалагна маанилеринде $1+\sin x+\sin x^2+...+\sin x^n$ эсептегиле.
8.	1 ден 100 гө чейинки натуралдык сандарды тескери тартипте ирети менен экранга чыгар

№17 ЛАБОРАТОРИЯЛЫК ИШ

Циклдын үч формасын тең колдонуп маселелердин программасын түзүү

Иштин максаты: Циклдын үч формасын тең колдонуп маселелердин программасын түзүү менен окуучулардын мурда алган билимдерин калыптандыруу

Түшүндүрмө

Мисалы: $y=x^2+5x-4$ функциясынын маанисин x тин 1 ден 10 го чейинки маанилеринде эсептегиле.

1. "Азырынча" циклинде карайлы.

Паскаль тилинде:	Бейсик тилинде:

program esep;
var x,y:integer;
begin x:=1;
while x<=10 do begin
y:=sqr(x)+5*x-4;
writeln('x=',x,','y=',y); x=x+1; end;
end.
	REM
	x=1
	10 IF x>=10 THEN 20
	y=x^2+5*x-4
	PRINT x=';x,'y=';y x=x+1
	GOTO 10
	20 END

2. "Чейин" циклинде карайлы:

Паскаль тилинде:	Бейсик тилинде:

program esep;
var x,y:integer;
begin x:=1;
repeat
y:=sqr(x)+5*x-4;
writeln('x=',x,' ','y=',y);
x=x+1;
until x>=10;end.
	REM
	x=1
	10 y=x^2+5*x-4
	PRINT 'x=';x,'y=';y
	x=x+1
	IF x<=10 THEN 10
	END

3. Параметрлүү циклде карайлы:

Паскаль тилинде:	Бейсик тилинде:

program esep;
var x,y:integer;
begin
for x:=1 to 10 do begin
y:=sqr(x)+5*x-4;
writeln('x=',x,' ','y=',y);
end;end.
	REM
	FOR x=1 TO 10 STEP 1
	y=x^2+5*x-4
	PRINT 'x=';x,'y=';y
	NEXT x
	END

Төмөнкү тапшырмаларды аткаргыла(Паскаль жана Бейсик тилдеринде):

1.	n натуралдык сандын суммасынан квадраттык тамыр чыгаруу программасын түз.
2.	Удаалаш берилген k сандын квадраттарынын көбөйтүндүсүн эсептеген программа түз.
3.	Берилген z натуралдык сандарды тескери тартипте жайгаштыр жана алардын ар бирин k-даражага көтөрүү программасын түз.
4.	(2n-1)!·(3n-2)! ни эсептеген программа түз.
5.	$t = ay^2 + by + c$ функциясынын маанилерин n натуралдык сандар үчүн эсептегиле, таблицасын түзгүлө

Бейсик тилиндеги параметрлүү циклди колдонуп төмөнкү тапшырмаларды аткаргыла

6.	i нин мааниси улам k га чоңоёт деп эсептеп, берилген функциянын маанисин эсептегиле: $d = bi\frac{i + ci^2 - \sqrt{ci}}{i+5}$
7.	a_1, ... , a_{50} анык сандардын удаалаштыгы берилген. Эгер удаалаштыктын ар бир мүчөсү кийинки турган мүчөдөн n эсе чоң болсо, анда бул удаалаштыктын суммасын эсептегиле.
8.	Берилген k сандардын арасынан 3 кө так бөлүнгөн сандардын суммасын эсептегиле.

№18 ЛАБОРАТОРИЯЛЫК ИШ

Саптык чоңдуктар менен иштөө.

Иштин максаты: Окуучуларды саптык чоңдуктар менен иштөөдөгү өзгөчөлүктөрдү кенен пайдаланууга багыттоо.

Түшүндүрмө:

Литердик чоңдуктар Паскаль тилинде STRING сапчасы менен берилет. Ал программанын аталыш аймагында өзгөрмөлөрдүн типтерин баяндоо бөлүгүндө жайгашкан. Мисалы:Var fd:string;

Ал эми программанын аткарылышында саптык өзгөрмөгө берилген сап аппострофко алынып менчиктелин жазылат:

fd:='Бишкек';

Паскаль тилинде саптар менен төмөнкүдөй аракеттерди аткара алабыз:

1. Саптарды бириктирүү операциясы(конкетанация). Бул операция +(бул математикадагы кошуу белгиси эмес) белгиси менен белгиленет.
2. Салыштыруу операциялары: =; >=; >; <>; <; <=. Ар кандай узундуктагы саптарды салыштырууга болот. Салыштыруу ASCII кодуна ылайык солдон оңду көздөй жүргүзүлөт.
3. Саптын белгиленген бөлүгүн көчүрүп алуу - copy(st,n,k) функциясы, мында st - көрсөтүлгөн сап, n - баштапкы позиция, k - көчүрүү үчүн зарыл болгон символдордун саны
4. Саптын белгиленген бөлүгүн өчүрүү – delete(st,n,k) процедурасы, мында st – көрсөтүлгөн сап, n - баштапкы позиция, k – өчүрүлүүчү символдордун саны
5. Көрсөтүлгөн сапка белгиленген саптын бөлүгүн коюу – insert(st1,st,n)процедурасы, мында st1 - белгиленген саптын болүгү, st - көрсөтүлгөн сап, n - ордуна кою башталуучу символдордун позициясы.

6. Белгиленген саптын бөлүгүн көрсөтүлгөн саптан издоо – k:=pos(st1,st) функциясы, мында st1 - белгиленген саптын бөлүгү, st – көрсөтүлгөн сап, k– белгиленген саптын бөлүгүнө алгачкы кирүү позициясы
7. k:=length(st); функциясы саптын узундугун көрсөтөт.
8. STR процедурасы каалаган анык же бүтүн типтеги санды символдор сабына айландырат: str(b,st);
   Мында: b - сан өзгөрмөсү, st - символдор сабы
9. VAL процедурасы цифраларды камтыган сапты анык же бүтүн типтеги санга айландырат: val(st,b,code);

Бейсик тилинде саптык чоңдуктарды айырмалап белгилөө максатында аларды A$, B$, ж.б. катарында белгилешет. Бул саптык өзгөрмөгө берилген сап тырмакчага(") алынып менчиктелип жазылат: A$="Бишкек"

Бейсик тилинде саптык чоңдуктар менен иштөөдө төмөнкү операторлорду колдонобуз:

1. Саптын узундугу LEN(A$) операторунун жардамы менен аныкталат.
2. Саптарды бириктирүү операциясы(конкетанация). Бул операция +(бул математикадагы кошуу белгиси эмес) белгиси менен белгиленет.
3. Салыштыруу операциялары: =; >=; >; <>; <; <=. Ар кандай узундуктагы саптарды салыштырууга болот. Салыштыруу ASCII кодуна ылайык солдон оңду көздөй жүргүзүлөт.
4. Саптын ортосунан кандайдыр символдордун удаалаштыгын кесип алуу MID$(A$,N,K) операторунун жардамы менен берилет. Мында A$ – берилген сап, N – баштапкы позиция, K – кесип алуу үчүн зарыл болгон символдордун саны
5. Саптын оң жагынан кесип алуу RIGHT(A$,K) оператору менен коштолот: Мында K – кесип алуу үчүн зарыл болгон сим-волдордун саны
6. Саптын сол жагынан кесип алуу LEFT(A$,K) оператору менен коштолот: Мында K – кесип алуу үчүн зарыл болгон сим-волдордун саны

Мисалы: "Кыргызстан – көп улуттуу мамлекет" деген саптан "Кыргызстан", "мамлекет", "көп улуттуу" деген сөздөрдү алгыла. Ошондой эле "Орто Азиядагы " деген сөздү берилген сапка кошуп жазгыла. "Кыргызстан – мамлекет" деген жаңы сүйлөмдү алгыла.

Паскаль тилинде:

```
program sap;
uses crt;
var a,b,c,d,s,l:string[50];k,t:integer;
begin
```

```
clrscr;
a:='Кыргызстан - коп улуттуу мамлекет';
writeln(a);t:=length(a);
writeln('саптын узундугу - ',t);
b:=copy(a,1,10);writeln(b);t:=length(b);
writeln('саптын узундугу - ',t);
c:=copy(a,26,8);writeln(c);t:=length(c);
writeln('саптын узундугу - ',t);
d:=copy(a,14,11);writeln(d);t:=length(d);
writeln('саптын узундугу - ',t);
s:='Орто Азиядагы ';writeln(s);t:=length(s);
writeln('саптын узундугу - ',t);
l:=s+a; writeln(l);t:=length(l);
writeln('саптын узундугу - ',t);
delete(a,13,12);writeln(a );
t:=length(a);writeln('саптын узундугу - ',t);
insert(s,a,14);writeln(a);t:=length(a);
writeln('саптын узундугу - ',t);writeln;
writeln('"Орто Азиядагы" деген сапты издоо');
k:=pos(s,a);
writeln('"Орто Азиядагы" д-н сап-н позициясы-',k);
end.
```

Бейсик тилинде:

```
REM
CLS
A$="КЫРГЫЗСТАН - КОП УЛУТТУУ МАМЛЕКЕТ"
B=LEN(A$):PRINT A$:PRINT "САПТЫН УЗУНДУГУ-";B
C$=LEFT$(A$,10):PRINT C$
B=LEN(C$):PRINT "САПТЫН УЗУНДУГУ-";B
D$=RIGHT$(A$,8): PRINT D$
B=LEN(C$):PRINT "САПТЫН УЗУНДУГУ-";B
F$=MID$(A$,14,11): PRINT F$
B=LEN(F$):PRINT "САПТЫН УЗУНДУГУ-";B
G$="ОРТО АЗИЯДАГЫ": PRINT G$
B=LEN(G$):PRINT "САПТЫН УЗУНДУГУ-";B
H$=C$ +" - "+" "+G$+" "+D$
PRINT H$:B=LEN(H$):PRINT "САПТЫН УЗУНДУГУ-";B
K$=G$+" "+A$: PRINT K$
B=LEN(K$):PRINT "САПТЫН УЗУНДУГУ-";B
END
```

Төмөнкү тапшырмаларды аткаргыла:

1.	Берилген сүйлөмдүн узундугун эсептегиле жана ал сүйлөмдү түзгөн сөздөрдү чыгаргыла.
2.	Берилген саптагы “а” тамгаларынын санын эсептегиле.
3.	Берилген саптарды салыштыргыла.
4.	Берилген сөздөрдөрдөн сүйлөм түзгүлө.
5.	“2001-жыл – туризм жылы” деген саптан “туризм” деген сөздүн позициясын аныктагыла.
6.	“Кыргызстанда жети область бар. Алар: Чүй, Ош, Жалал-Абад, Нарын, Талас, Баткен, Ысык-Көл” деген саптан “Кыргызстандагы областтар: Чүй, Ош, Жалал-Абад, Нарын, Талас, Баткен, Ысык-Көл” деген сапты алгыла.
7.	Берилген саптан “р”, “н”, “о” тамгалары кездешкен сөздөрдүн санын эсептегиле жана ал сөздөрдү чыгаргыла.
8.	Берилген саптан бардык “ы” тамгаларын өчүргүлө.

№19 ЛАБОРАТОРИЯЛЫК ИШ

Бир өлчөмдүү массивдер менен иштөө

Иштин максаты: Бир өлчөмдүү массивдер катышкан амалдарды аткаруу менен окуучулардын билимин тереңдетүү

Түшүндүрмө

Бир өлчөмдүү массивдерди баяндоо Паскаль тилинде төмөндөгүдөй жүргүзүлөт:

a:array[1..n] of <массивдин тиби>;

мында а– массивдин ысмы; n– массивдин элементтеринин саны; мисалы: a:array[1..6] of real; Паскаль тилинде массив чарчы кашаага “[“, “]” алынып жазылат. Мисалы: a[n], b[7], g[100], ж.б.

Ал эми Бейсик тилинде массивди баяндоо үчүн DIM оператору колдонулат: DIM A(N),мында А – массивдин ыс-мы; N – массивдин элементтеринин саны; Бейсик тилинде массив тегерек кашаага “(“, “)” алынып жазылат.

Паскаль жана Бейсик программалоо тилдеринде бир өлчөмдүү массивдер менен иштөө үчүн цикл уюштурулат.

Мисалы: B(10)={2, 5, -6, 30, -3, 3, 6, 1, -45, 50} бир өлчөмдүү массиви берилген. Массивдин оң элементтерин суммасын, терс элементтеринин кобойтундусун аныктаган программа түзгүлө.

Паскаль тилинде:

```
program mas_1;
var   b:array[1..10] of real;i:integer;s,p:real;
begin
 writeln('массив-н элемент-н киргиз');
 for i:=1 to 10 do readln(b[i]);
```

```
 for i:=1 to 10 do writeln(b[i]);
 s:=0; p:=1; for i:=1 to 10 do
 if b[i]>0 then s:=s+b[i] else p:=p*b[i]
 writeln('s=',s:8:3);writeln('p=',p:8:3);
end.
```

Бейсик тилинде:

```
REM
DIM b(10)
PRINT "массивдин элементтерин киргиз"
FOR i=1 TO 10:INPUT b(i): PRINT b(i):NEXT i
s=0: p=1:FOR i=1 TO 10
IF b(i)>0 THEN s=s+b(i) ELSE p=p*b(i)
NEXT i: PRINT "s=";s, "p=";p
END
```

Төмөнкү тапшырмаларды аткаргыла:

1.	$s_1, s_2, \ldots, s_{20}$ анык сандарынан турган бир өлчөмдүү массив берилген. Бул массивдин элементтерин төмөнкү тартипте жайгаштыргыла: $s_{20}, s_{19}, \ldots, s_1$
2.	$k_1, k_2, \ldots k_{16}$ анык сандарынан турган бир өлчөмдүү массив берилген. Бул массивдин элементтерин төмөнкү тартипте жайгаштыргыла: $k_1, k_9, k_2, k_{10}, \ldots, k_8, k_{16}$
3.	$c_1, c_2, \ldots c_{18}$ бүтүн сандарынан турган бир өлчөмдүү массив берилген. Бул массивдин элементтерин төмөнкү тартипте жайгаштыргыла: $c_1, c_{18}, c_2, c_{17}, \ldots, c_8, c_9$
4.	$z_1, z_2, \ldots z_{10}$ бүтүн сандарынан турган бир өлчөмдүү массив берилген. Бул массивдин элементтерин төмөнкү тартипте жайгаштыргыла: $c_1+c_{10}, c_2+c_9, \ldots, c_5+c_6$
5.	G(N) бир өлчөмдүү массивинин максималдуу жана минималдуу элементтеринин суммасын эсептегиле.
6.	E(M) бир өлчөмдүү массивиннин элементтерин өсүү тартибинде жайгаштыргыла.
7.	P(K) бир өлчөмдүү массивинин элементтери кемүү тартибинде жайгаштыргыла.
8.	$d_1, d_2, \ldots d_{14}$ бүтүн сандарынан турган бир өлчөмдүү массив берилген. Бул массивдин элементтеринен төмөнкүнү алгыла: $\max(d_1+d_{14}, d_2+d_{13}, \ldots, d_7+d_8)$
9.	X(K) бир өлчөмдүү массивинин оң элементтерине a санын көбөйтүп C(Z) массивине менчиктегиле, терс элементтеринен b санын алып салып, O(E) массивине мен-чиктегиле.
10.	n натуралдык саны жана $q_1, q_2, \ldots q_n$ анык сандарынан турган бир өлчөмдүү массив берилген. Эгер n 2 ге так бөлүнсө, анда бул массивдин элементтеринен төмөнкүнү алгыла: $\min(q_1+q_2, q_2+q_3, \ldots, q_{n-1}+q_n)$

№20 ЛАБОРАТОРИЯЛЫК ИШ

Эки өлчөмдүү массивдер менен иштөө

Иштин_максаты: Эки өлчөмдүү массивдер катышкан амалдарды аткаруу менен окуучулардын билимин тереңдетүү

Түшүндүрмө

Эки өлчөмдүү массивдерди баяндоо Паскаль тилинде төмөндөгүдөй жүргүзүлөт:

A:array[1..N,1..M] of <массивдин тиби>;

Ал эми Бейсик тилинде массивди баяндоо төмөндөгүдөй жүргүзүлөт: DIM A(N,M)

мында A – массивдин ысмы; NxM – массивдин элементтеринин саны; N – массивдин сапчаларынын саны; M – массивдин мамычаларынын саны

Паскаль тилинде: a:array[1..3,1..4] of real;

Бейсик тилинде: DIM A(3,4)

Паскаль жана Бейсик программалоо тилдеринде эки өлчөмдүү массивдер менен иштөө үчүн камтылган циклдар, б.а. сапча боюнча өзүнчө цикл, мамыча боюнча өзүнчө цикл уюштурулат.

Мисалы: C(N,N) эки өлчөмдүү массиви берилген. Саптардын биринчи элементинин өсүү тартиби боюнча массивдин саптарын иреттегиле. N=4 деп алгыла.

Паскаль тилинде:

```
program iret;
const n=4;
type sapcha=array[1..n] of real;
     matrix=array[1..n] of sapcha;
var v:sapcha; c:matrix; i,j,k:integer;
begin
    for i:=1 to n do
    for j:=1 to n do     read(x[i,j]);
    for i:=1 to n do begin
    for j:=1 to n do
    write(x[i,j]:6:1,' '); writeln; end;
    writeln;
    for i:=1 to n do begin k:=i;
    for j:=i+1 to n do
    if x[j,1]<x[k,1] then k:=j;
    v:=x[i]; x[i]:=x[k]; x[k]:=v;
    for j:=1 to n do
    write(x[i,j]:6:1,' '); writeln;
    end;  end.
```

Бейсик тилинде:

```
REM
n=4 :DIM v(n),x(n,n)
FOR i=1 TO n: FOR j=1 TO n
INPUT x(i,j):NEXT j:NEXT i
FOR i=1 TO n: FOR j=1 TO n
PRINT x(i,j);:NEXT j:PRINT:NEXT i:PRINT
FOR i=1 TO n: k=i:FOR j=i+1 TO n
IF x(j,1}<x{k,1) THEN k=j:NEXT j
FOR j=1 TO n
v(i)=x{i,j}: x{i,j}=x(k,j): x(k,j)=v(i)
NEXT j:NEXT i
FOR i=1 TO n:FOR j=1 TO n: PRINT x(i,j);
NEXT j:PRINT:NEXT i
END
```

Төмөнкү тапшырмаларды аткаргыла:

1.	A(N,M) эки өлчөмдүү массивинин оң элементтеринин суммасын жана терс элементтеринин көбөйтүндусүн эсептегиле.
2.	Берилген S(K,L) эки өлчөмдүү массивинин 1-сапчасынын элементтерин акыркы сапчанын элементтери менен алмаштыргыла.
3.	Берилген A(P,H) эки өлчөмдүү массивинин максималдуу элементи турган сапчасын чыгаргыла.
4.	Берилген A(T,M) жана B(Q,R) эки өлчөмдүү массивдеринин элементтеринин суммаларын жана көбөйтүндүлөрүн салыштыргыла
5.	Берилген D(N,N) эки өлчөмдүү массивинин элементтеринин арасынан окшош элементтер болсо, анда ал элементтердин жайгашкан ордун көрсөткүлө
6.	Квадраттык матрицанын негизги диогоналынын жогору жагында жаткан элементтердин максималдуу элементин тапкыла
7.	Квадраттык матрицанын негизги диогоналынын төмөн жагында жаткан элементтеринин минималдуу элементин тапкыла.
8.	Квадраттык матрицаны транспонирлегиле.
9.	Берилген эки өлчөмдүү массивдин элементтерин ичинен 3 кө так бөлүнгөндөрүнүн санын эсепте.
10.	Берилген эки өлчөмдүү массивдин ар бир сапчасынын максималдуу элементин тап жана алардан бир өлчөмдүү маасивди түзгүлө.

№21 ЛАБОРАТОРИЯЛЫК ИШ

Камтылган программаны уюштуруу

Иштин__максаты: Камтылган программаны уюштуруу менен окуучулардын билимин тереңдетүү.

Түшүндүрмө:

Паскаль тилинде камтылган программаларды уюштуруу үчүн атайын процедуралар жана функциялар колдонулат. Камтылган программа негизги программанын аталыш аймагында, өзгөрмөлөрдүн типтери баяндалгандан кийин жайгашат. Процедуранын аталыш аймагы procedure кызматчы сөзүн жана процедуранын ысмын камтыйт, алар Паскаль тилинин эрежелерине баш ийишет. Процедуранын жалпы жазылыш форматы төмөнкүдөй:

```
procedure proc(<кызматчы маалымат - параметрлер>)
var {локалдуу өзгөрмөлөрдү баяндоо бөлүгү}
    begin
    ...{камтылган программаны түзгөн операторлордун
    удаалаштыгы}
    end;
```

Ал эми функциялардын аталыш аймагы жана түзүлүшү төмөнкүлөрдү өз ичине алат:

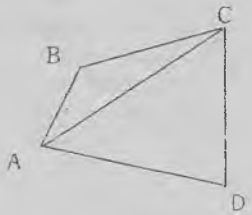
```
function<ысмы>(формалдуу параметрлер тизмеси)
var {локалдуу өзгөрмөлөрдү баяндоо бөлүгү}
    begin
        ... {функциянын түзүлүшү}
    end;
```

Негизги программадан камтылган программага кайрылуу процедуранын жана функциянын ысымдары боюнча чакырылат.

Бейсик тилинде камтылган программаны уюштурулушу GOSUB N жана RETURN операторлорунун жардамы менен жүргүзүлөт жана негизги программанын каалаган жеринде жайгашат. Аны негизги программага чакыруу үчүн GOSUB N операторун колдонобуз. Мында N – камтылган программанын башталыш сабында турган номер. Ал RETURN оператору менен аяктайт. Бул болсо RETURN оператору аткарылгандан кийин кайра негизги программага кайрылуу дегенди билдирет.

Мисалы: Жактары жана диогоналы берилген томпок төрт бурчтуктун аянтын тапкыла.

Сүрөттөн көрүнүп тургандай диогонал томпок төрт бурчтукту эки үч бурчтукка бөлөт. Албетте, бул үч бурчтуктарга Герондун формуласын колдонуп, алардын аянтын эсептеп



алабыз. Анда томпок төрт бурчтуктун аянты бул үч бурчтуктардын суммасы болуп эсептелет.

Героидун формуласы: $S=\sqrt{p(p-a)(p-b)(p-c)}$. Мында a,b,c – үч бурчтуктун жактары, $p=\frac{a+b+c}{2}$ – үч бурчтуктун жарым периметри. Көрүнүп тургандай биз Герондун формуласын эки жолу эсептешибиз керек. Биз бул маселени жөнөкөйлөтүп, камтылган программанын жардамы менен чыгаралы.

Паскаль тилинде(процедуранын жардамы менен):

```
program ayant;
var AB,BC,CD,DA,AC,s1,s2,s3:real;
procedure trian(var a,b,c,s:real);
      var p:real;
      begin
      p:=(a+b+c)/2;s:=sqrt(p*(p-a)*(p-b)*(p-c));
      end;
begin
read(AB,BC,CD,DA,AC);
trian(AB,Bc,Ac,s1);trian(CD,DA,AC,s2);
s3:=s1+s2;write('аянт=',s3:8:3);
end.
```

Паскаль тилинде(функциянын жардамы менен):

```
program ayant;
var AB,BC,CD,DA,AC,s:real;
function trian(a,b,c:real):real;
      var p:real;
      begin
      p:=(a+b+c)/2;trian:=sqrt(p*(p-a)*(p-b)*(p-c));
      end;
begin
read(AB,BC,CD,DA,AC);
s:= trian(AB,BC,AC)+trian(CD,DA,AC);
write('аянт=',s:8:4);
end.
```

Бейсик тилинде:

```
REM
GOSUB 140
k=s
GOSUB 140
q=s:s1=k+q
PRINT "аянт=";s1
END
140 INPUT a,b,c
```

p=(a+b+c)/2:s=SQR(p*(p-a)*(p-b)*(p-c))
RETURN

Төмөнкү тапшырмаларды аткаргыла:

1.	x, y, z сандары берилген. Эки сандын чоңун табууну камтылган программа катарында карап, бул сандардын чоңун тапкыла
2.	n натуралдык саны жана $a_1,\ldots,a_n$ анык сандарынын удаалаштыгы берилген. Эгерде $x = a_1 \cdot a_2 \cdot \ldots \cdot a_n$, $y = a_{n+1} \cdot a_{n+2} \cdot \ldots \cdot a_{2n}$, $z = a_{2n+1} \cdot a_{2n+2} \cdot \ldots \cdot a_{3n}$ болсо, анда $x+y^2+z^3$ ту эсептегиле.
3.	Үч бурчтуктун аянтын эсептөөнү колдонуп берилген n бурчтуктун аянтын эсептегиле.
4.	Анык a жана b сандары берилген. Төмөнкүнү эсептегиле: u=min(a,b); v=min(ab, a+b); min(u+v²,3.14)
5.	Составы 5 адамдан турган командаңы 8;10;11 кандидаттын арасынан тандап алуунун канча ыкмасы бар? Саноо ыкмаларын камтылган программа түрүндө көрсөткүлө
6.	Футболист 1 м бийиктиктен топту 20 м/с баштапкы ылдамдык менен вертикалдуу жогору тебет. 1 с, 2 с, 4с убакыттан кийин топ кандай бийиктикте болот? Бийиктикти эсептөөнү функциянын жардамы менен жүргүзгүлө
7.	1,2,3,4,5 цифралары менен берилген бардык үч орундуу сандардын арасынан, 2,4,6,8 цифралары менен берилген бардык эки орундуу сандардын арасынан, 1,3,7,8,9 цифралары менен бардык төрт орундуу сандардын арасынан максималдуу маанисин тапкыла. Туура келген сандарды эсептөөнү камтылган программа түрүндө көрсөткүлө.
8.	Темир жол вокзалына бир күндө орто эсеп менен 5 поезд келет. Күнүнө 2 поезд, 4 поезд, 6 поезд келеринин ыктымалдуулугун тапкыла. Ыктымалдуулукту эсептөөнү камтылган программа түрүндө көрсөткүлө

№22 ЛАБОРАТОРИЯЛЫК ИШ

Программалоо тилинде графика менен иштөө.

Иштин максаты: Окуучуларга ар кандай чиймелерди компью-тердин жардамы менен сызууга үйрөтүү

Түшүндүрмө

Паскаль тилинде сызыктарды чийүү үчүн алгач графикалык режимди орнотуп алышыбыз керек. Бизге белгилүү болгондой Uses Graph модулун орнотобуз, ал эми программанын бөлүгүндө милдетүү түрдө:

gd:=detect; initgraph(gd,gm,'');

жазылышы керек. Эгер программа түзүүдө жогорку кадамдар иштелбсе, программаны компиляциялоодо Error 15:File not found(Graph.tpu) маалыматы чыгат. Бул графикалык режим туура эмес орнотулду дегенди билдирет.

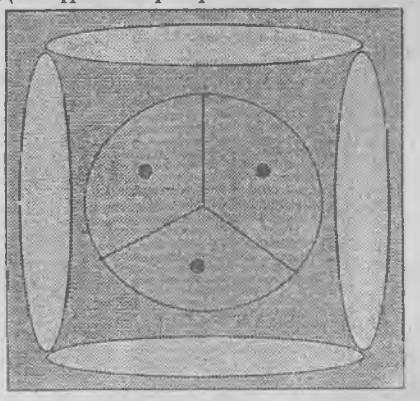
Паскаль тилинде жөнөкөй фигурала төмөнкү процедуралар менен берилет:

Экрандын фонунун түсү –SetFillStyle(Style,Color)
Сызыктын түсү – SetColor(Color)
Чекит – PutPixel(x,y,C)
Кесинди – Line(x1,y1,x2,y2)
Тик бурчтук – Rectangle(x1,y1,x2,y2)
Боёлгон тик бурчтук – Bar(x1,y1,x2,y2)
Параллелепипед – Bar3D(x1,y1,x2,y2, Depth,Top)
Айлана – Circle(x,y,R)
Айлананын жаасы – Arc(x,y:,StAngle,EndAngle,R)
Айлананын боёлгон сектору – PieSlice(x,y,St,Sp,R)
Эллипс – Ellipse(x,y,StAngle,EndAngle,Rx,Ry)
Боёлгон эллипс – FillEllipse(x,y:,Rx,Ry)

Бейсик тилинде графикалык режимде иштөөдө SCREEN N режимин орнотушубуз зарыл. Бул режимге карата экрандагы сүрөттөлүштөр берилет. Бейсик тилинде жөнөкөй фигуралар төмөнкү операторлордун жардамы менен сызылат:

Чекит – PSET(x,y),C
Кесинди – LINE(x1,y1)-(x2,y2),C
Тик бурчтук – LINE(x1,y1)-(x2,y2),C, B
Боёлгон тик бурчтук – LINE(x1,y1)-(x2,y2),C, BF
Айлана – CIRCLE(x,y),R,C
Эллипс – CIRCLE(x,y),R,C, , ,U
Айлананын жаасы – CIRCLE(x,y),R,C,Q,W,U
Боё – PAINT(x,y),C

Мисалы: Экранга жогоруда келтирилген командалар катышкан төмөндөгүдөй сүрөттөлүштү сызалы:



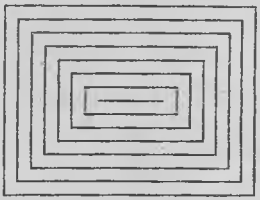
Паскаль тилинде:

```
program graph;
uses graph;
var gd,gm,errcode:integer;
begin
gd:=detect;initgraph(gd,gm,'');errcode:=graphresult;
if errcode=grOk then begin
setfillstyle(1,3);bar(180,90,480,370);
setcolor(6);ellipse(330,120,0,360,120,15);
ellipse(330,340,0,360,120,15);
ellipse(210,230,0,360,15,110);
ellipse(450,230,0,360,15,110);
setcolor(5);circle(330,230,80);
setcolor(6);line(330,150,330,230);
line(330,230,256,260);line(330,230,403,260);
setcolor(2);putpixel(285,210,9);
putpixel(375,210,9);putpixel(330,275,9);
setfillstyle(1,5);fillellipse(330,120,120,15);
fillellipse(330,340,120,15);
fillellipse(210,230,15,110);
fillellipse(450,230,15,110);
setfillstyle(1,8);pieslice(330,230,0,360,80);
closegraph;
end; else writeln('error');
end.
```

Бейсик тилинде:

```
REM:SCREEN 9
LINE(140,50)-(510,300),11,BF
CIRCLE(325,80),140,13,,,.12:PAINT(325,80),13
CIRCLE(325,270),140,13,,,.1:PAINT(325,270),13
CIRCLE(185,175),93,13,,,4.2:PAINT(185,175),13
CIRCLE(465,175),93,13,,,4.2: PAINT(465,175),13
CIRCLE(325,175),90,4:PAINT(325,175),4
LINE(325,110)-(325,175),3:LINE(325,175)-(410,200),3
LINE(325,175)-(240,200),3
PSET(285,160),1:PSET(365,160),1:PSET(325,210),1
END
```

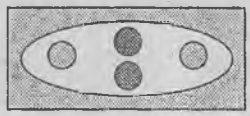
Төмөнкү тапшырмаларды аткаргыла:

1. 

6. 

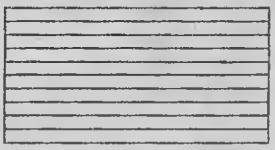
2. 

7. 

3. 

8. 

4. 

9. 

5. 

10. 

№23 ЛАБОРАТОРИЯЛЫК ИШ

Файлдар менен иштоо

Иштин максаты: Окуучуларга файлдар менен иштөөнү үйрөтүү менен билимдерин калыптандыруу.

Түшүндүрмө

Паскаль тилинде типтешкен файлдарды баяндоо төмөнкүчө жүрөт:

```
type fil=file of integer;
var f:fil;
```

Дискте файл түзүү үчүн төмөнкү процедуралар аткарылышы зарыл:

- assign(f,'<дисктеги файлдын ысмы>');
- дисктеги файлды ачуу: rewrite(f);
- файлдын компонентин жазуу: write(f,a)

Мында f - файлдык өзгөрмөнүн ысмы, а - буфердик өзгөрмө, анын мааниси дискке камтылат. а өзгөрмөсүнүн тиби милдеттүү түрдө файлдын компонентинин тиби менен бирдей болушу керек.

- close(f) – файлды жабуу

Эгерде мурда дискте жазылган файлды окуу керек болсо, анда аракеттердин удаалаштыгы төмөнкүдөй болот:

- assign(f,'<дисктеги файлдын ысмы>');
- окуу үчүн файлды ачуу: reset(f);
- файлдын компонентасын окуу: read(f,a); Мында f - файлдык өзгөрмөнүн ысмы, а - буфердик өзгөрмө, ага файлдык өзгөрмө камтылат.
- close(f) – файлды жабуу

Файлдын керектүү болгон компонентасына көрсөткүчтү жылдыруу үчүн seek процедурасы колдонулат:

seek(<файлдык өзгөрмөнүн ысмы>,n);

n- компонентанын номери, ал бүтүн типте болот.

Файлдын өлчөмүн аныктоо:

t:= filesize(<файлдык өзгөрмөнүн ысмы>);

t - бүтүн сандагы өзгөрмө

Көрсөткүчтүн жайгашкан ордун аныктоо үчүн Турбо-Паскалда filepos функциясы колдонулат:

filepos(<файлдык өзгөрмөнүн ысмы>);

Мисалы: Берилген symbol символдордун файлынан бардык кичине латын тамгаларын чыгаруу.

```
Program tamga;
type v=file of char;
var symbol:v; s:char;
begin
assign(symbol,'sum.txt');reset(symbol); while not eof(symbol) do begin
read(symbol,s);
if (s<='z') and (s>='a') then writeln(s); end; end.
```

Көрүнүп тургандай symbol өзгөрмөсү мурда берилген sum.txt файлынын файлдык өзгөрмөсү болуп эсептелет.

Бейсик тилинде файлдар менен иштөөдө төмөнкү операторлор колдонулат:

Файлды ачууда OPEN оператору колдонулаарын жогоруда айтып өттүк. Жалпы жазылыш форматы төмөнкүдөй:

OPEN <файл$> FOR <режим> ACCESS <жол> <жабуу> AS # <файлдын номери> LEN =<жазылыштын узундугу>

<файл$>– файлдын ысмы. Ал түзүлүштү же файлга баруу жолун камтышы мүмкүн.

<режим>–APPEND,BINARY,INPUT,OUTPUT,RANDOM режим-деринин бири. Булар жөнүндө 3-бөлүмдөгү "Файлдар менен иштөө" темасынан кеңири тааныша аласыңар.

<жабуу> AS – тармактык түз аракет үчүн файлдын жабылуу шартын аныктайт: SHARED(жалпы), LOCK READ (окуу үчүн жабуу), LOCK WRITE(жазуу үчүн жабуу), LOCK READ WRITE(окуу-жазуу үчүн жабуу)

<файлдын номери> – ачылган файлды идентификациялоочу 1 ден 255 ке чейинки сан.

LEN =<жазылыштын узундугу>–түз аракеттеги файлдар үчүн жазылыштын узундугу(128 байт); удаалаш файлдар үчүн буфердеги символдордун саны(512 байт)

Мисалы:
```
OPEN "TEST.DAT" FOR OUTPUT AS #1
PRINT #1, "Текст":CLOSE
OPEN "TEST.DAT" FOR INPUT AS #1
PRINT INPUT$(5, 1)    'биринчи беш символду чыгарат
CLOSE
```

Төмөнкү тапшырмаларды аткаргыла:

1.	Символдук f файлы берилген. Бул файлдын көчүрмөсүн g файлында алгыла.
2.	f1 жана f2 символдук файлдары берилген. f1 файлынын компоненттерин f2 файлына көчүрүп келгиле.
3.	"Класс" деген файл түзүп, ага окуучулардын фамилиясын, атын, туулган жылын, айын, күнүн файлдын компоненттери катары жазгыла.
4	f1 символдук файлы берилген. Бул файлдын компоненттерин f2 файлын тескери тартипте жазгыла.
5.	"Чейрек" деген файл түзгүлө. Анын компоненттери катары предметтер, окуучулардын фамилиялары, алган баалары болуп эсептелинсин.
6.	Футболдук чемпионаттын таблицасын файл катары түзгүлө. Мында файлдын компоненттери командanын аты, ойногон оюндарынын саны, утканы, тең чыкканы, утулганы, алган упайы болуп саналат.
7.	Балдар алма теришти. Ар бир баланын бир күндө терген алмаларынын салмагы боюнча файл түзгүлө
8	Сатуучунун бир күндө саткан товарларынын санын жана алган кирешесин файл катарында чагылдыргыла.

№24 ЛАБОРАТОРИЯЛЫК ИШ

Татаал графикалык сүрөттөлүштөр менен иштөө
(кошумча окуу үчүн)

Иштин максаты: Окуучуларга жалаң гана графикалык процедуралардын(операторлордун) жардамы менен эмес, Паскаль (Бейсик) тилинин башка мүмкүнчүлүктөрүн колдонуп татаал сүрөттөлүштөрдү сызууну үйрөтүү.

Мисалы: Төмөндөгү программа экранда вертолеттун кыймылын сүрөттөйт. Оңго, солго, жогору жана төмөн баскычтарынын жардамы менен вертолеттун кыймылын башкара аласыңар.

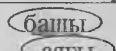
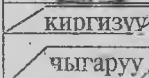
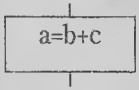
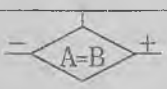
Программа Паскаль тилинде түзүлгөн.

```
Program GraphHelecopter;
uses Graph,Crt;
Var  PCursor:Pointer; Ch:Char; TheEnd:Boolean;
 x,y,xn,yn,i,GraphDriver,GraphMode,ErrorCode:Integer;procedure
MyGraphInit;
begin GraphDriver := Detect;
 InitGraph(GraphDriver,GraphMode,'');
 ErrorCode:=GraphResult; if ErrorCode<>grOk then begin
 Writeln (GraphErrorMsg (ErrorCode));
 Writeln (Программанын иштеши үзгүлтүккө учурады!');
 Halt(1); end; end;
procedure CursorDraw (var x,y:Integer;var p: Pointer);
const Cursor:array[1..10] of PointType=((x:210;y:404),
                (x:215; y:409),(x:225; y:411),(x:230; y:422),
                (x:180; y:422),(x:170; y:415),(x:160; y:414),
                (x:160;y:410),(x:190; y:410),(x:190; y:404));
var Size:Word;
begin SetFillStyle(9,7);FillEllipse(200,400,20,4);
 FillEllipse(154,412,6,6); SetFillStyle(1,3);
 DrawPoly (10,Cursor); FloodFill(200,420,White);
 Size:=ImageSize (148,396,230,422);
 GetMem(p,Size);GetImage(148,396,230,422,p^);
 x := 148; y := 396;
end;
procedure NewXY (var x,y:Integer;var Flag:Boolean);
const Esc=#27;LeftArrow=#75;RightArrow=#77;
    UpArrow=#72;DownArrow=#80;
var ArrowsAndEsc:set of char;
   ExtendedKey:Boolean; Ch:Char;
begin ArrowsAndEsc:=[LeftArrow,RightArrow,UpArrow,
    DownArrow, Esc ];
```

```
repeat ExtendedKey:=False;Ch:=ReadKey;
 if Ch=Esc then Flag:=True;if Ch=#0 then begin
 Ch:=ReadKey;ExtendedKey:=True; end;
 if ExtendedKey then
 case Ch of
 LeftArrow: x:=x-10; RightArrow:x:=x+10;
 UpArrow:  y:=y-10; DownArrow: y:=y+10;
 end;
if x<0 then x:=GetMaxX-85;if x>GetMaxX-85 then x:=0;
if y<0 then y:=0;if y>GetMaxY-30 then begin
  SetColor(15);SetFillStyle(1,15);
  for i:=0 to 46 do begin FillEllipse(x+45,y+15,i,i);
   Delay(30);Sound(random(5000)); end;
  SetColor(0);SetFillStyle(1,0);
  for i:=0 to 46 do begin FillEllipse(x+45,y+15,i,i);
   Delay(30);Sound(random(5000));end; NoSound;
  SetColor(4);SetTextStyle(DefaultFont,HorizDir,2);
  outtextxy(130,200,'Учканды үйрөн'); Delay(7000);
  Flag:=True;end;
until Ch in ArrowsAndEsc; end;
begin MyGraphInit;
 Line(0,getmaxy-2,getmaxx,getmaxy-2);
 CursorDraw(x,y,PCursor); xn:=x; yn:=y;TheEnd:=False;
 while KeyPressed do Ch:=ReadKey;
 repeat NewXY (xn, yn, TheEnd);
  PutImage(x,y,PCursor^,XorPut);
  PutImage(xn,yn,PCursor^,XorPut);x:=xn;y:=yn;
  Delay(100);
  until TheEnd;
CloseGraph;
end.
```

ТИРКЕМЕЛЕР

1. Блок-схемада колдонулуучу геометриялык фигуралар

Шарттуу белгилер	Блокторго түшүндүрмө	Аткарган кызматы
	Алгоритмдин башы жана аягы.	Блок схеманын башталышын же аякташын көрсөтөт.
	Киргизүү, чыгаруу блогу.	Берилген маанини берилген чоңдукка жазат же андан чыгаруучу түзүлүштөргө чыгарат.
	Иштеп чыгаруучу блок(Арифметикалык блок).	1. Блоктун ичине жазылган туюнтманы эсептейт. 2. Барабардыктын сол жагына жазылган чоңдуктун маанисин туюнтманын маанисине алмаштырат.
	Логикалык блок. Бул блоктун ичине шарт жазылат.	Блоктун ичине жазылган шарт орун алса, анда башкаруу(ооба) "+" багытка, антпесе (жок) "–" багытка жиберилет.
	Түшүнүк берүүчү блок.	Бул чийме ага туташтырылган блок эмне кызмат аткараарын түшүндүрөт.
	Аныкталган процессти көрсөтүүчү блок.	Алдын ала аныкталган процессти көрсөтөт.
	Модификация жасоочу блок.	Циклдин параметрин өзгөртүү дегенди билдирет.
	Документ	Натыйжаны кагазга чыгарат.
	Дисплей	Эсептин берилишин же натыйжаны дисплейге чыгарат же андан башка түзүлүшкө жиберет.
	Туташтыруучу	Эки алыс жайгашкан блокту туташтырат.
	Багыт көрсөтүүчү сызыктар.	Бир блок иштегенден кийин 2-кайсы блок иштерин көрсөтөт.

2. Turbo Pascalдын функционалдык баскычтары

F1	Жардам
F2	Түзүлгөн же түзөтүү киргизилген файлды диске жазуу
F3	Дискте жазылган программалык текстти түзөтүү киргизүү же иштетүү үчүн окуу
F4	Программаны курсор турган сапка чейин аткаруу.
F5	Учурдагы терезечени толук экранга чагылдыруу же тескерисинче.
F6	Келерки терезечени учурдагы терезече менен алмаштыруу.
F7	Кадамдап иштөө, программанын иштешинде курсор турган саптан кийин программанын иштешин улантат.
F8	Программанын иштешинде курсор турган саптан кийин программанын иштешин улантат.
F9	Программаны иштетпестен текшерүү гана болот.
F10	Негизги меню сапка чыгуу.
Ctrl-F3	Программалык стектик терезечени ачуу
Ctrl-F1	Контекстик маалымат алуу.
Ctrl-F2	Программанын иштешин токтотуу.
Ctrl-F9	Программанын тексттик жазылышын текшерип аны оперативдик эске сактоо менен бирге программаны иштетип, андан соң паскаль чөйрөсүнө келет.
Ctrl-F4	Туюнтманы эсептөө же өзгөрмөнү өзгөртүү.
Ctrl-F5	Терезечени жайгаштыруу.
Ctrl-Del	Буферди тазалоо.
Ctrl-Ins	Буферге көчүрүү.
Alt-C	Compile менюсуна кирүү
Alt-D	Debug менюсуна кирүү
Alt-E	Edit менюсуна кирүү
Alt-H	Help менюсуна кирүү
Alt-O	Options менюсуна кирүү
Alt-R	Run менюсуна кирүү
Alt-S	Search менюсуна кирүү
Alt-W	Window менюсуна кирүү.
Alt-X	Паскалдан чыгуу.
Alt-F1	Маалымдамаларды чакыруу (маалымдама).
Alt-F3	Учурдагы терезечени жабуу.
Alt-F5	Программалык терезечени көрүү.
Alt-F9	Программаны текшерүү.
Shift-F1	Маалымдамалардын тизмеси.
Shift-F6	Алдыдагы терезечени алуу.
Shift-Del	Редактордогу блокту буферге алуу.
Shift-Ins	Буферден редакторго көчүрүү.

3. Qbasicтин функционалдык баскычтары	
F1	Жардам
F2	Сакталган файл жана модулдар менен иштөө
F3	REM операторуна курсорду алып келүү
F4	Программанын аткарылыш жыйынтыгын көрүү
F5	Программаны аткаруу
Alt-F	File менюсуна кирүү
Alt-E	Edit менюсуна кирүү
Alt-V	Viev менюсуна кирүү
Alt-S	Search менюсуна кирүү
Alt-R	Run менюсуна кирүү
Alt-D	Debug менюсуна кирүү
Alt-O	Options менюсуна кирүү
Alt-H	Help менюсуна кирүү
Shift-Del	Редактордогу блокту буферге алуу.
Shift-Ins	Буферден редакторго көчүрүү.
Shift-F5	Программаны аткаруу
Ctrl-Ins	Буферге көчүрүү.

Текстти редактирлөө(оңдоо)

Курсордун жылышы			
Ctrl-S	бир символ солго	Ctrl-Page Down	программанын аягына
Ctrl-A	бир сөзгө солго	Ctrl-Page Up	программанын башына
Ctrl-X	бир сапка төмөн	Ctrl-C же Page Down	бир баракка төмөн
Ctrl-D	бир символ оңго	Ctrl-End	Экрандын аягына
Ctrl-F	бир сөзгө оңго	Ctrl-Home	Экрандын башына
Ctrl-E	бир сапка жогору	Ctrl-R же Page Up	бир баракка жогору
Home	саптын башына	End	саптын аягына

Кыстаруу \өчүрүү	
Ins же Ctrl-V	кыстаруу режимин тактоо (өчүрүү же күйгүзүү)
Ctrl-N	санты кыстаруу
Ctrl-Y	санты өчүрүү
Ctrl-Q,Y	курсор турган позициянын оң жагын өчүрүү
Ctrl-T	курсор турган позициянын оң жагындагы бир сөздү өчүрүү

Адабияттар:

1. Эшенкулов П.А. "Информатика жана эсептөөчү техниканын негиздери", 1997-ж.
2. Эшенкулов П.А. "Бейсик тилин көнүгүү жана мисалдар менен үйрөнүү", 2000-ж.
3. Грызлов В.И., Грызлова Т.П. "Турбо Паскаль 7.0", 2000-ж.
4. Ларинова Информатика и вычислительная техника, 1992 г.
5. Б.Д. Баячорова, А.С Өмүралиев "Эсептөөчү машиналар, информатика жана программалоо I бөлүк, 1988-ж
6. Г. И. Светозарова, А.А. Мельников, А.В. Козловский "Практикум по программированию на языке Бейсик", 1992 г.

МАЗМУНУ

Прогаммалоонун негиздери

Бектенова Д. Б., Кыштобаева Т. Т., Молдошев Р. А, Асанова М. Б, Мокешов Ж. К.

Улуттук компьютердик гимназия.
Биздин дарек: Токтогул көчөсү 68
☎ 28-44-72
URL: www. ncg.edu. kg
e-mail: office@ncg.edu.kg