**1. АЛГОРИТМДЕУ НЕГІЗДЕРІ**

* 1. **Негізгі ұғымдар мен түсініктер**

ЭЕМ-ді пайдалану істерін қарастырмас бұрын оның жұ- мысымен тығыз байланысты алгоритм, программа ұғымдарын білуіміз қажет. Əрбір ЭЕМ алдын ала берілген алгоритммен, яғни жоспармен жұмыс істейді. Алгоритмді заңдылық, реттел- ген амалдар жиыны, кезекпен орындалатын операциялар тізімі деп ұғынған жөн. Бұл ұғым қазіргі кезде кеңінен қолданылып жүр. Оның көптеген анықтамалары да бар. Соның бірін келтіре кетейік.

**Алгоритм** – *берілген есептің шығару жолын реттелген амал- дар тізбегі түріне келтіру*. Кез келген есепті қарапайым амал- дарды тізбектей орындау арқылы шығаруға болады. Алгоритмді ЭЕМ-де орындау үшін оны программа түрінде жазып шығу керек. Сонымен, алгоритм оны атқарушы ЭЕМ-ге жұмыс тəртібін түсіндіретін ережелер мен нұсқаулар тізбегінен тұрады. Алгори- тмді атқарушының рөлін негізінен адам немесе автоматтан- дырылған аспап, яғни ЭЕМ, робот, т.б. атқарады. Мысалы, *y=(ax+b)(cx-d)* функциясын есептеу төменгі іс-əрекеттерден

тұрады:

1. *а*-ны *х*-ке көбейту, оны R1 деп белгілеу;
2. оған *b*-ны қосу, нəтижесін R2 деп белгілеу;
3. *с*-ны *х*-ке көбейту, оны R3 деп белгілеу;
4. одан *d*-ны алу, оны R4 деп белгілеу;
5. R2-ні R4-ке көбейту, оны *y* деп белгілеу.

Алгоритмнің орындалу кезінде оны орындаушыға келесі жолы

қандай нұсқау бойынша орындалатыны белгілі болуы қажет. Ал орындаушының жүзеге асыра алатын командалар жиыны – ко- мандалар жүйесін құрайды.

Алгоритм мен программаға байланысты ЭЕМ-нің мынадай жұмыс ерекшеліктері болады:

1. есепті шығару жолы алгоритм түрінде өрнектелуі қажет;
2. алгоритм программаға айналдырылуы тиіс;
3. программа машина жадына енгізіліп, ретімен орындалуы керек. Алгоритм күнделікті тұрмыста да кеңінен қолданылады. Мысалы, студент болу үшін алгоритмнің мынадай қадамдарын орындау керек.
4. Орта мектепті бітіріп, аттестат алу.
5. Керекті құжаттарды аттестаттың түпнұсқасымен бірге белгілі бір оқу орнына өткізу.
6. Конкурстан өту.

Бұл көрсетілген пункттердің орнын ауыстыруға болмайды. Олар көрсетілген ретпен кезектесіп орындалуы тиіс. Сонда ғана керекті нəтижеге (студент болу) қолымыз жетеді.

Алгоритм информатиканың іргелі ұғымдарының бірі. Квадрат теңдеудің түбірін табу ережесі, үшбұрыштың ауданын есептеу жолдары алгоритмдердің мысалдары болып табылады.

**Алгоритмдеу** – *есепті шығару алгоритмін құрастыру про- цесі*, мұның нəтижесінде мəліметтерді өңдеу процесінің кезең- дері айқындалады да, кезеңдер мазмұны формальды (жасанды) түрде жазылып, солардың орындалу реттілігі анықталады.

**Алгоритмдік тіл** – алгоритмдерді жазуға арналған символ- дар мен сол символдардан тұратын конструкцияларды құрас- тыру жəне түсіндіру ережелерінің жиыны.

**Программалау тілі** ЭЕМ-дерде программаларды орындау ісін атқарады.

**Программа** *– алгоритмді машинаға түсінікті нұсқаулар тізімі ретінде жазу.* Программада берілген мəліметтердің си- паттамаларымен бірге оларды өңдейтін командалар болады. Осы командалар тізбегі орындалу барысында есептің нəтижесі шығады. Командалар қандай мəліметтер қандай операцияларға қатынасатынын, олар қандай реттілікпен орындалатынын жəне нəтиженің қандай түрде шығарылатынын көрсетеді. Бұлар опера- торлар арқылы жүзеге асырылады.

Əрбір ЭЕМ алдын ала жазылған программамен істейді. Про- цессор программаның құрамындағы командаларды кезекпен орындап отырады. Командалар тізбегін программа деп қарас- тыруға болады. Команда бір ғана қарапайым амалды орындау үшін берілген бұйрық ретінде беріледі. Командалар: арифметика- лық немесе логикалық амал; ақпаратты тасымалдау командасы;

берілген сандарды салыстыру командасы; келесі командаларға көшу тəртібін орындау, т.с.с.

Алгоритм жəне программа ұғымдары ұқсас екені көрініп тұр, алгоритм есептің шығару жолын қарапайым əрекеттер тізбегімен өрнектесе, программа сол өрнекті машинаға түсінікті тілмен жазып береді. Сондықтан программа жазу тəсілдері солардың қандай ережені пайдаланып, қандай компьютерлерге арналып жазылғанына байланысты алгоритмдік тілдер немесе программа- лау тілдері ұғымына келіп тіреледі.

**Мəліметтер** – белгілі бір процесс көмегімен тасымалдап, өңдеуге болатын, формальды түрде бейнеленген фактілер мен идеялар.

**Оператор** – операциялар мен мəндерді көрсететін немесе солардың элементтерінің қай жерде орналасқанын білдіретін символдар жиыны. Мысалы:

А:= В+С; {А, В, С – айнымалы;} К:= 2; IF T<0 THEN ...

**Айнымалы** – программа орындалуы барысында өз мəнін

өзгерте алатын объект.

# Айнымалы қасиеттері:

1. айнымалы белгілі бір мəнге ие болғанша, анықталмаған болып саналады. Оған мəн беру мынадай тəсілдермен орындалады:

* сырттан енгізу арқылы;
* тұрақты мəнді (константаны) меншіктеу арқылы;
* бұрын анықталған айнымалының мəнін беру арқылы;

1. кез келген сəтте айнымалының белгілі бір мəні болады не- месе ол анықталмаған болып есептеледі;
2. айнымалыға соңғы берілген мəн оның алдыңғы мəнін жой- ып (өшіріп) жібереді. Айнымалыны таңдау (оқу) жəне оны пайдалану айнымалының мəнін өзгертпейді.

Бұл пəннің заттық негізі болып (ЭЕМ-де шығару мақсатын- да) алгоритмдер мен программаларды құрастыру тəсілдері мен құралдары саналады. Программалар құру үшін программалау жүйелері пайдаланылады.

**Программалау жүйесі** – программалауды автоматтандыру құралдары. Олар программалау тілінен, осы тілдің трансляторы-

нан, құжаттамаларынан жəне де программаларды дайындау, əрі орындау құралдарынан тұрады.

**Транслятор** – бір тілді екінші тілге аудару программасы. Ол интерпретатор жəне компилятор сияқты екі топқа бөлінеді.

**Интерпретатор** – бұл командаларды аударып, оларды бірден орындауға арналған трансляторлық программа.

**Компилятор** – бұл алгоритмдік тілдің конструкцияларын толығымен машиналық кодқа түрлендіретін программа. Есептің нəтижесін алу түшін машиналық кодты орындау керек.

* 1. **Алгоритм қасиеттері**

Алгоритм ұғымының мəнін ашатын негізгі қасиеттерінен немесе оған қойылатын талаптардан қысқаша мағлұматтар кел- тірейік. ЭЕМ-де орындалуға тиіс алгоритмдерге мынадай талап- тар қойылады:

1. ол анық əрі дəл өрнектелуі тиіс – детерминділік (анықтылық, бір мəнділік) қасиеті, яғни алгоритмді басқаша түсінуге жол бер- мей, тек қана көрсетілген əрекеттерді айқын түрде орындауға арналған нұсқаулар дəлдігі;
2. алгоритм шектелген уақыттан соң нəтиже беруі тиіс – нəтижелілік қасиеті, мұнда белгілі бір əрекеттер саны атқарылған соң, процестің қажетті нəтижесін алып, оны аяқтау мүмкіндігі- нің болуы немесе есептеу процесін ары қарай жалғастыруға болмайтындығы жайлы мəлімет алуымыз қажет;
3. бір тектес есептерге жалпы бір ғана алгоритм қолданылуы тиіс – жалпылық қасиеті, ол алгоритмнің осы сияқты көптеген басқа да есептерге қолданылу мүмкіндігінің болуын көрсетеді;
4. алгоритмді кішкене бөліктерге бөлу мүмкіндігі болуы қажет – модульдік (дискреттілік, бөліктік) қасиет – есептеу процесін жекеленген қарапайым операцияларға бөлу мүмкін- дігінің болуы, яғни күрделі есепті атқарылуына күдік келтіруге болмайтын шағын бөліктерге жіктеу орындалуы тиіс.

Біріншіден, алгоритм анық, əрі *дəл өрнектелуі* қажет. Онда қандай қадамдар көрсетілсе, тек соны ғана орындау керек. Есеп шығару жолына керектің бəрі біржақты анықталуы жəне орындаушыға түсінікті, əрі нақты болуы тиіс. Екіншіден, алго- ритм *нəтижелі болуы* керек. Əрекеттердің шектелген санынан

кейін белгілі бір уақыт ішінде қорытынды нəтиже алуымыз қажет. Əрбір алгоритм біршама бастапқы мəліметтердің болуын талап етеді жəне іздеген нəтижені алуға жеткізеді. Мысалы, сандарды қосу алгоритмі үшін бастапқы мəліметтерге қосылғыштар мəні жатады, ал нəтижесі қосынды болады. Үшіншіден, алгоритмнің *жалпылық қасиеті* болады, яғни бастапқы мəліметтер мəнінің бір жиыны бір ғана нəтиже береді. Егер берілген мəліметтер өзгерсе, нəтиже де өзгереді. Басқаша айтқанда, бір алгоритм бір типтес есептердің əр түрлі алғашқы мəліметтері үшін əр түрлі нəтижелер беруі тиіс. Мысалы, квадрат теңдеуді шешу алгоритмі кез келген *а*, *b*, *с* мəндері үшін оның түбірін дұрыс табуы керек. Төртіншіден, алгоритмнің *үзік-үзік модульдерге бөліну* қасиеті болуы тиіс, яғни үлкен алгоритмді бірнеше кішкене алгорит- мдерге жіктеуге əрқашанда мүмкіншілік болуы керек. Сондықтан алгоритмді екі-үш бөлікке бөліп, оларды өзінше құра алатын дəрежеде жұмыс істелуі қажет. Олар тек бірінің қорытындысын келесі жолы керекті мəлімет ретінде қолдануы тиіс.

* 1. **Алгоритмдерді бейнелеу жолдары**

Алгоритмдерді ЭЕМ-де орындау үшін оларды алдын ала жазып алу керек, яғни ол белгілі бір заңдылықпен өрнектелуі тиіс. Жалпы алгоритмді жазып өрнектеу, яғни оларды бейнелеу түрлеріне төмендегі тəсілдер жатады:

* табиғи тіл сөздері арқылы;
* формулалық-сөздік тəсіл арқылы;
* графикалық түрде бейнелейтін блок-схемалар арқылы;
* псевдокодтар арқылы;
* құрылымдық диаграммалар арқылы;
* программалау тілі арқылы.

Алгоритмдерді **табиғи тіл сөздері арқылы** бейнелеуде – есептеу кезеңдері мазмұны кез келген түрде табиғи тілде жазы- лады.

Осы тəсілмен келесі мысалдың алгоритмін жазып шығайық. Сандар жиымы (массиві) берілген делік. Осы жиым сандарының көрсетілген аралықта, яғни интервалда толығынан жататынын/ жатпайтынын тексеру керек. Интервал өзінің шекаралық А жəне В мəндерімен берілген.

1. Бірінші санды аламыз.
2. Осы сан интервалға кіретінін салыстыру жолымен тексереміз; егер жауабы «Иə» болса, онда **3-пунктке көшу**, əйтпесе (жау- абы – «Жоқ» болса) – **6-пунктке көшу**.
3. Жиымның барлық элементтері қарастырылды ма? Егер жауа- бы «Иə» болса, онда **5-пунктке көшу**, жауабы «Жоқ» болса,

# – 4-пунктке көшу.

1. Келесі элементті қарастырамыз. **2-пунктке көшу**.
2. Мынадай хабарлама шығару: барлық элементтер осы интер- валға кіреді. **7-пунктке көшу**.
3. Мынадай хабарлама шығару: элементтер интервалға толы- ғынан кірмейді.
4. Соңы.

Бұл тəсілде көрнекілік жоқ, яғни толық формальдау мүмкін- дігі жоқ. Жалпы алгоритмді табиғи тілде өрнектеу ЭЕМ-дерде қолданылмайды, өйткені онда дəлдік, нақтылық болмайды.

Алгоритмдерді **формулалық-сөздік тəсіл арқылы** бейне- ленуі – тапсырманың математикалық символдар мен өрнектер- дің жəне сөздердің араласуымен берілуі болып табылады.

Мысалы, үшбұрыш ауданын оның үш қабырғасының ұзын- дығы арқылы есептеу алгоритмін құру керек болсын делік.

1. – үшбұрыштың жарты периметрін есептеу

*p*=(*a+b+c*)/2

1. – үшбұрыштың ауданын есептеу

*S =*

*p(p* − *a*)( *p* −*b*) *p* −*c*

1. – нəтиже ретінде *S* мəнін шығарып, алгоритм жұмысын аяқтау. Бұл тəсілді пайдаланғанда, алгоритмді кез келген деңгейде айқындап көрсетуге болады, бірақ формальды түрде анық бейне-

леу қиын.

Алгоритмді **графикалық түрде блок-схемалар арқылы** көрсету – оның логикалық құрылымын графикалық түрде бей- нелеу болып саналады. Бұл – алгоритмдерді өрнектеудің ең көп тараған түрі. Мұнда мəліметтерді өңдеудің əрбір кезеңі атқарыла- тын операцияға сəйкес əр түрлі геометриялық фигуралар (блок- тар) түрінде көрсетіледі.

Əр блоктың ішіне орындалатын іс-əрекеттің (амалдың) маз- мұны жазылады. Символдардың (блоктардың) бір кіру жəне бір шығу сызықтары болуға тиіс.

Графикалық жолмен алгоритмдерді жазу үшін мемлекеттік стандарт белгіленген, онда кез келген амал белгілі бір геомет- риялық фигурамен өрнектеледі. Ол фигуралар немесе блоктар амалдар символы деп те аталады. Блоктар бағытталған сызық- тармен байланысып, бірінен соң бірі орналасады. Жиі қолданы- латын амалдар, яғни мəліметтерді ЭЕМ-ге енгізу, формуламен есептеу, шарттардың орындалуын тексеру, нəтижені қағазға басу символдары 1-кестеде көрсетілген. Осы суреттегі көрсетілген блоктардан (символдардан) алгоритм схемалары құрастырыла- ды. Алгоритмдер схемасымен ақпаратты өңдеудің əрбір сатысы немесе орындалатын операциялар реті анықталады. Кейде алго- ритмдер схемасын оның блок-схемасы деп те атайды.

* 1. ***кесте***

**Алгоритмдерді бейнелеу блоктары**

|  |  |  |
| --- | --- | --- |
| **Іс-əрекеттің аты** | **Блоктың пішімі** | **Атқаратын**  **жұмысы** |
| Процесс |  | Математикалық өрнектерді есептеу |
| Басы – соңы |  | Алгоритмдерді бастау, аяқтау |
| Алдын ала анықталған процесс (подпрограмма) |  | Қосалқы программаларға кіру жəне шығу |

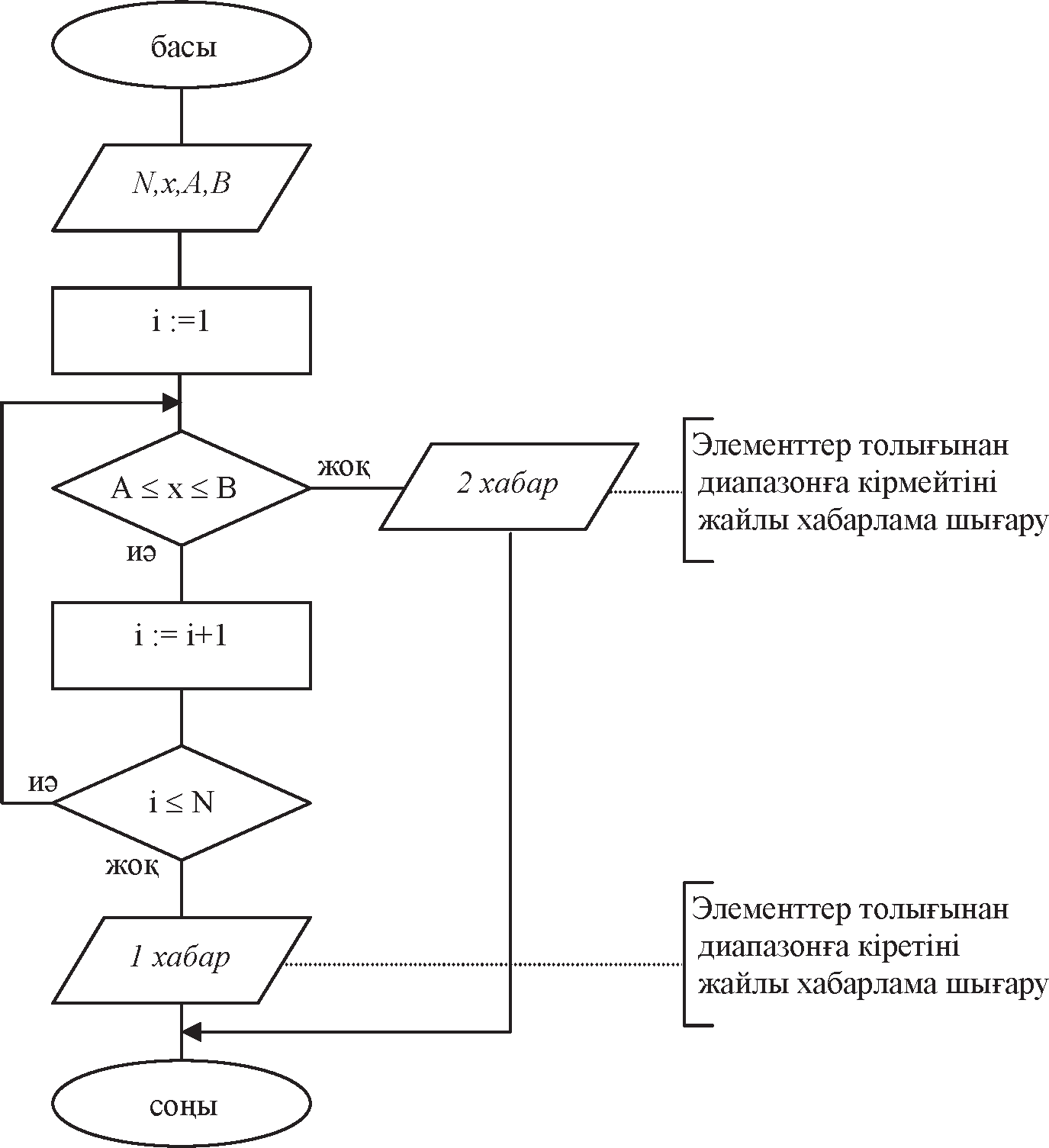
|  |  |  |
| --- | --- | --- |
| Шешім |  | Есеп шығару жолын таңдау |
| Модификация |  | Цикл басы |
| Құжат |  | Нəтижені баспаға (қағазға) шығару |
| Енгізу-шығару |  | Мəліметтерді енгізу жəне шығару |
| Түсініктеме |  | Схеманы, формула- ларды түсіндіру |

Сонымен алгоритм блоктармен немесе геометриялық көпбұ- рыштар түріндегі фигуралармен өрнектеледі.

Блок-схемалар дəстүрлі (кəдімгі) жəне құрылымды болып екіге бөлінеді.

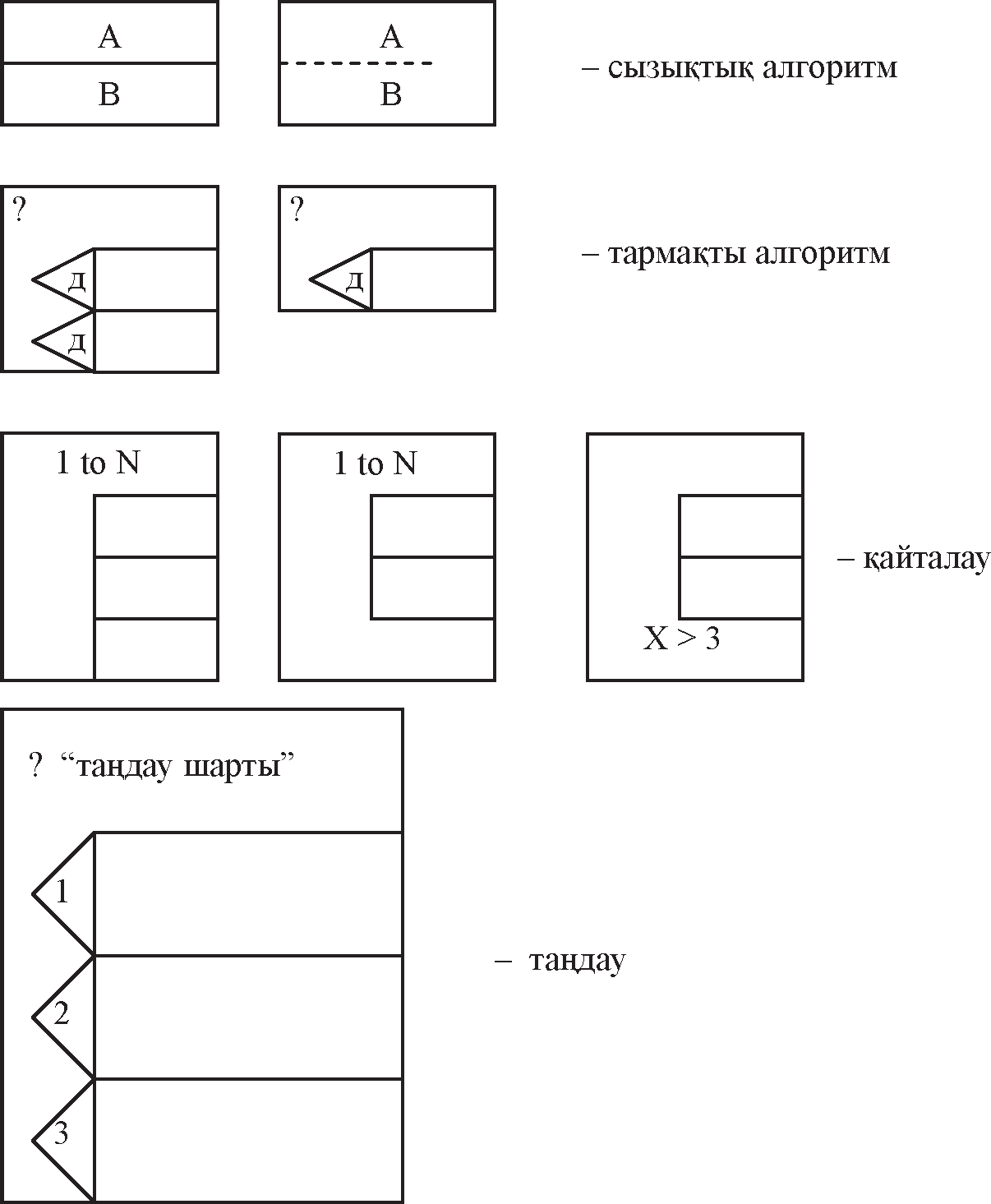
Табиғи тіл сөздері арқылы көрсетілген алгоритмдегі мысал- ды блок-схемалар арқылы бейнелеу төмендегі суретте көрсетіл- ген. Онда берілген жиым сандарының көрсетілген аралықта,

яғни интервалда толығынан жататынын/жатпайтынын тексеру керек болатын. Интервал өзінің шекаралық А жəне В мəндерімен берілген.



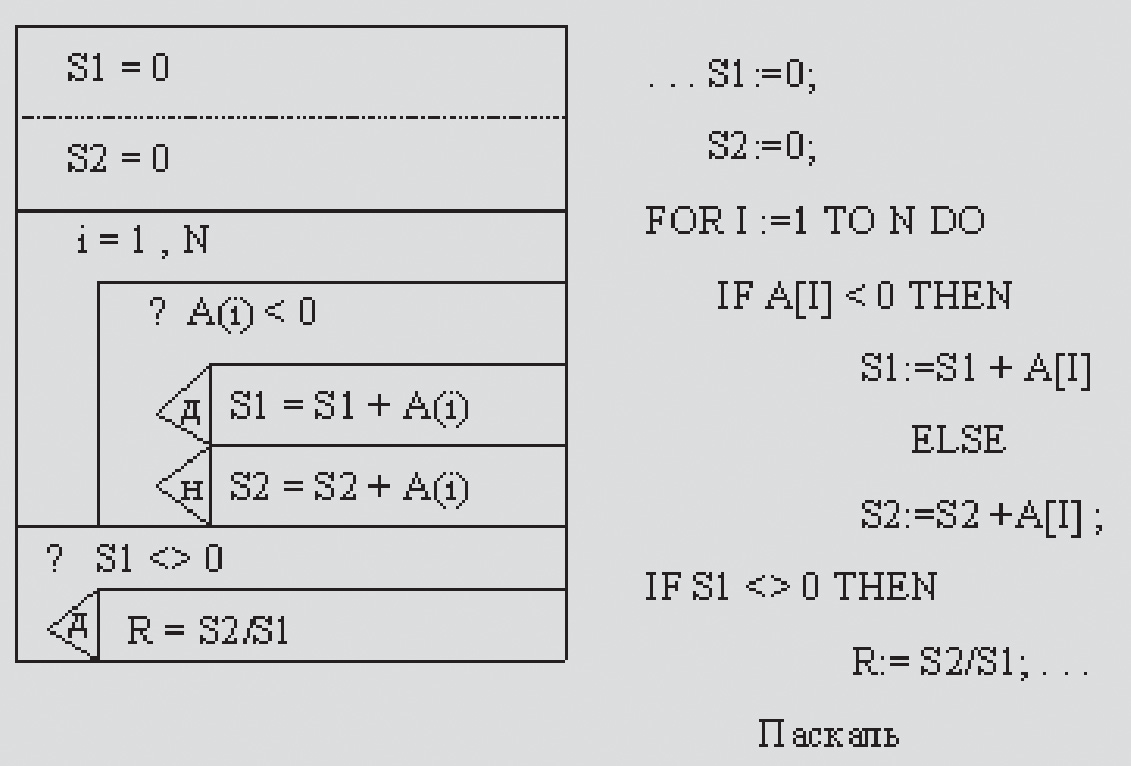
**Құрылымдық диаграммалар** – модульдер арасындағы бай- ланысты көрсететін, мəліметтер құрылымын, программаларды жəне мəліметтерді өңдеу жүйелерін бейнелейтін құрылымдық блок-схемалар ретінде пайдаланылады. Құрылымдық диаграмма- лардың бірнеше түрлері болады: Насси-Шнейдерман диаграмма- сы, Варнье диаграммасы, Джексон диаграммасы, МЭСИД жəне т.б. диаграммалары.

# МЭСИД диаграммасының негізгі элементтері



МЭСИД диаграммасын пайдалануға бір мысал келтірейік. Оң жəне теріс сандардан тұратын бір өлшемді жиым беріл-

ген. Осы жиымдағы оң элементтердің қосындысын оның теріс элементтерінің қосындысына бөлгендегі бөліндіні анықтау қа- жет. Төменде осы есептің диаграммасы мен соған сəйкес Паскаль тілінің операторлары оның оң жақ бөлігінде келтірілген.



**Программалау тілдері –** программаларды ЭЕМ-де тікелей орындауға арналған алгоритмдерді жазу тəсілі. Программа – алгоритмнің ЭЕМ-ге түсінікті түрде жазылуы.

Əрбір машинаның өз тілі (машиналық тіл) болады жəне ол тек осы тілдегі программаларды, яғни командалар тізбегін орын- дай алады. Программаларды машиналық тілде жазу өте күрделі, əрі адамды шаршататын жұмыс болып табылады. Программа- лаушылардың жұмыс өнімділігін арттыру мақсатында жасанды тілдер, яғни программалау тілдері қолданылады. Мұндайда жа- санды тілде жазылған программа машиналық тілге аударылуы тиіс. Аудару жұмысын, яғни программаны бір тілден екінші тілге түрлендіру ісін транслятор атқарады. Жиі қолданылатын, тікелей интерпретация жасайтын транслятор түріне Бейсик тілінің транс- ляторы жатады, ол командаларды аударады да, оны бірден орын- дайды. Мұндай транслятор жұмысының қорытындысы қажетті нəтижелерді алу болып табылады.

Паскаль тілінің трансляторы – компилятор түрінде бола- ды. Мұнда бастапқы жазылған программа мəтіні машина тіліне аударылады да, *объектілік модуль* деп аталатын программа коды шығарылады. Сонан соң объектілік модуль *Программа аралық байланыс редакторы* деп аталатын программа арқылы өңдел- геннен кейін барып қана жұмыс істеуге дайын болады.

Алгоритмдік, яғни программалау тілдері есептерді шығару жолын баяндау-өрнектеу үлгісі, белгілі бір проблеманы шешу негізінде орындалатын əрекеттерге басшылық, ой еңбегін үнем- деуге мүмкіндік беретін əдіс, есеп шешімін табуды автомат- тандыруға қажетті іс-əрекет, жаңа проблеманы шешу кезінде қолданылатын тəсілдер, күрделі процестерді өрнектеу жəне мате- матикалық дəлдікпен анық етіп жазу құралы бола алады.

* 1. **ЭЕМ-де есеп шығару кезеңдері**

ЭЕМ-де есеп шығару күрделі процесс болып есептеледі, ол төмендегі кезеңдерден тұрады:

1. Берілген есепті математикалық түрде өрнектеу, яғни есепті мəселе ретінде қоя білу.
2. Есепті шығарудың ЭЕМ-ге ыңғайлы сандық тəсілдерін анықтау.
3. Есепті шығару жолын алгоритм түрінде бейнелеу.
4. Есепті ЭЕМ-де шығару программасын жасау жəне оның қа- телерін түзету.
5. Есепке керекті мəліметтер дайындау.
6. ЭЕМ-де есепті шығару жəне шыққан нəтижені іс жүзінде қолдану.

Берілген есепті математикалық түрде өрнектеу дегеніміз – есептің берілген мəндерін математикалық таңбаларды қолданып жаза білу жəне керекті математикалық формулаларды, өрнектерді анықтау болып саналады.

Күрделі формулаларды, теңдеулерді арифметикалық амалдар тізбегіне айналдыру есепті шығарудың сандық тəсілдерін табу не анықтау жолы болып есептеледі. Қазіргі кезде барлық есеп- тердің шығару жолының сандық тəсілдері белгілі десе де болады, тек солардың ішінен өзімізге тиімді жолын таңдап алуымыз ке- рек. Бұл мақсатта есепті шығару дəлдігін, нəтижені жылдам табу мүмкіндігін, мəліметтерді дайындау мен есепті шығарудың баға- сын салыстыра отырып қарастыру қажет.

Есептің алгоритмін жасағанда, оның шығару жолы тізбек- телген іс-əрекеттер ретінде схема түрінде өрнектеледі.

Программа жасағанда қазірде кеңінен тараған программа- лау тілінің бірінде алгоритм нақты түрде жазылады. Бізде кең

тараған тілдерге - Паскаль, Дельфи, Си жатады. Жазылған прог- рамманың қатесін түзету ЭЕМ-нің көмегімен шешіледі, өйткені жіберілген қателерді тек ЭЕМ ғана жылдам аңғарып, түзету мүмкіндігін береді.

Есепті шығаруға керекті деректерді сұрыпталған күйінде алдын ала қағазға, əйтпесе магниттік дискіге жазып, ЭЕМ-нің жадына реттей отырып енгіземіз. Есептің нəтижесін алған соң шешім қабылдау жəне оны іс жүзінде қолдану – мамандардың жұмысы. Тек солар ғана белгілі бір шешім қабылдай алады. Бірақ оқып-үйрену барысында кездесетін, яғни студенттерге арнал- ған есептерде жоғарыда көрсетілген сатылардың бірсыпырасы болмайды, өйткені олар бірден формула күйінде беріледі, шы- ғарудың сандық тəсілі формулада айқын көрініп тұрады (интеграл, туынды болмаса), нəтижені алған соң, оны жазып алу жеткілікті. Мəселені шешудің немесе есеп шығарудың көрсетілген алты са- тысы күрделі өндірістік есептерде, дипломдық немесе курстық жұмыстарда жиі кездеседі.

* 1. **Алгоритмдерді график түрінде жазу**

Алгоритмдер блоктардың өзара байланысуына қарай үш түрлі бірыңғай құрылымға – сызықтық, тармақтық жəне циклдік болып үш топқа бөлінеді. Енді соларды қарастырайық.

* 1. **Алгоритмдердің бірыңғай құрылымы**

Күрделі алгоритмдерді құру үшін қарапайым бірыңғайлан- ған алгоритмдік құрылымдар қолданылады. Олар сызықтық, тар- мақталу жəне цикл құрылымдарынан тұрады (2-кесте).

Программалау теориясында кез келген күрделі програм- маны үш түрлі құрылымнан құрастыруға болатыны дəлелден- ген, олар: *сызықтық*, *тармақты* жəне *циклдік* құрылымдар. Осы үшеуі құрылымдық программалаудың негізгі конструкциялары, яғни құраушылары болып саналады.

*Сызықтық құрылым* бірінен кейін бірі орындалып тізбек- теле орналасқан бірнеше операторлардан тұрады.

*Тармақты* – шартқа байланысты екі оператордың бірінің орындалуы

*Цикл* – операторлар бөлігінің бірнеше рет қайталана орын- далуы.

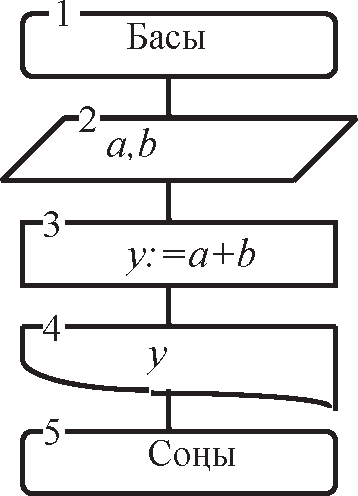
*Оператор* – тілдің қарапайым сөйлемі, ол белгілі бір əрекет немесе амал орындап, «**;»** таңбасымен аяқталады.

Негізгі конструкцияларды пайдалану мақсаты – қарапайым құрылымды программалау болып саналады. Мұндай программа- лар оңай оқылады, түзетіледі жəне керек болса, оңай өзгертіледі. Құрылымдық программалауда goto операторын қолдануға бол- майды, өйткені ол программа логикасын түсінуді қиындатады. Бірақ кейде goto операторын қолдану қажет болатын кездер болады.

* 1. ***кесте***

**Алгоритмдердің бірыңғай құрылымдары**

|  |  |  |
| --- | --- | --- |
| Сызықтық құрылым | Тармақты құрылым | Циклдік құрылым |
|  |  |  |



**1.1-сурет.** Алгоритм схемасы

Программа жұмысын басқару операторла- рын программаның басқарушы конструкция- сы деп атайды. Олар:

* құрама операторлар;
* таңдау операторлары;
* цикл операторлары;
* көшу операторы.

1. **Сызықтық құрылымды алгоритм** немесе қарапайым сызықтық алгоритм іс- əрекеттердің орындалу ретіне қарай тізбекте- ле орналасқан блоктардан тұрады. Амалдар- дың бұлай бірінен соң бірі реттеліп орындалу тəртібін табиғи атқарылу дейді.

Мысалы, *y= a+b* формуласы бойынша есептеу тіктөртбұрыш арқылы кескінделетін есептеу блогы (3-блок) арқылы өрнектеледі. Ал нəтижені қағазға басу үшін көпбұрышты құжат алу блогын (4-блок) пайдаланып, оның ішіне нəтиженің атауларын жазамыз. Жоғарыда көрсетілген *y=a+b* формуласымен есептеу үшін *a* əне *b*-ның сандық мəндерін ЭЕМ-ге енгізіп (2-блок), содан кейін қосу амалын орындап, ақырында *y*-ті қағазға басып шығарып, жұмысты тоқтатамыз. Осы алгоритмнің схемасы 1.1-суретте көрсетілген.

1. **Тармақталу алгоритмдері.** Тұрмыста кездесетін алгорит- мдер əр түрлі болып келеді. Олардың жиі кездесетін түріне алгоритмнің белгілі бір шарттың орындалуына не орындалмауы- на байланысты тармақталып бірнеше жолдарға бөлінуі жатады.

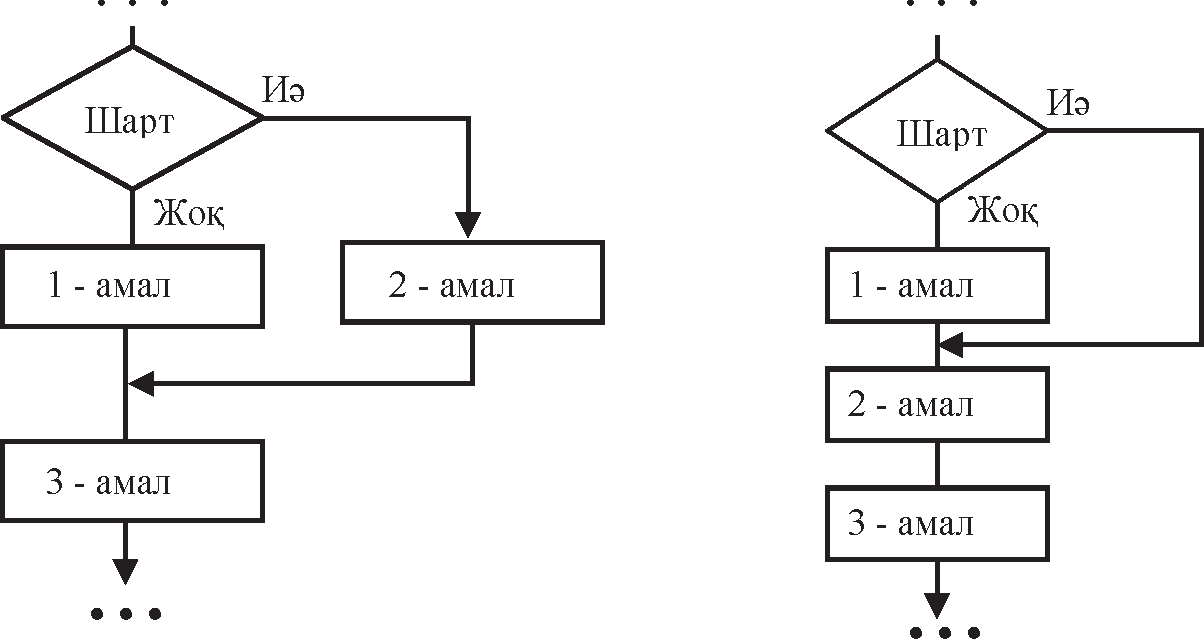
Тармақталу алгоритмінің құрылымы қарапайым болып ке- леді. Мұнда арифметикалық теңсіздік (теңдік) түрінде берілген логикалық шарт тексеріледі. Егер ол орындалса, онда алгоритм бір жолмен, ал орындалмаса екінші жолмен жүзеге асырыла- ды, яғни есепті шығару жолы тармақталып екіге бөлініп кетеді. Тармақталу алгоритмдеріне шартты тексеру блогы міндетті түрде кіреді. Ол ромб түрінде кескінделіп, басқа блоктармен 1 кіру жəне 2 шығу сызықтары арқылы байланысады. Көбінесе тармақталу алгоритмдері екі түрде кездеседі, олар «таңдау» жəне «аттап өту» мүмкіндіктерін іске асыруға көмектеседі.

«Таңдау» жолымен тармақталуда берілген шарт тексеріледі (1.2-сурет), егер ол шарт орындалса (орындалуы ақиқат бол- са), онда 2-амал жүзеге асырылып, содан кейін келесі 3-амалға көшеміз. Ал, егер де шарт орындалмаса, яғни оның орындалу мүмкіндігі жалған болса, онда 1-амал атқарылып, содан кейін 3-амал атқарылады. Сонымен, шарттың ақиқат немесе жалған бо- луына байланысты 1-амал немесе 2-амал орындалады.

«Аттап өту» (1.3-сурет) алгоритмінде шарт орындалса, 1-амалды аттап өтіп, бірден 2-амалды, содан кейін 3-амалды орындаймыз. Ал шарт жалған болса, онда 1-амал міндетті түрде орындалып, одан кейін 2- жəне 3-амалдар жүзеге асырылады. Тармақталу кезеңінде шартты тексеру блогы орындалуы ба- рысында, алгоритмнің екі мүмкіндігінің тек біреуі ғана таңдап

2–1618 **17**

алынып жүзеге асырылады да, ал екінші таңдап алынбаған тар- мақ біріктіру нүктесіне дейін орындалмай қалады. Енді осыған нақты мысалдар келтірейік.



**1.2-сурет.** «Таңдау» алгоритмі **1.3-сурет.** «Аттап өту» алгоритмі

*1-мысал. y* функциясын төмендегі формула бойынша есептеп шығару керек.

*y*  



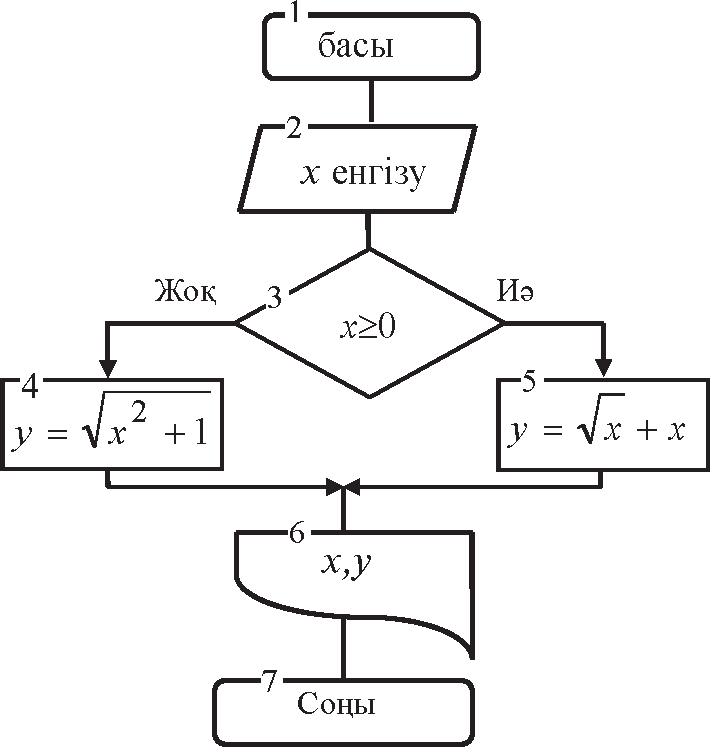


*x*  *x*, *x* 2  1,

*x*  0

*x*  0

Мұнда *x* айнымалысының

**1.4-сурет.** Тармақталу алгоритмі

таңбасына (оң, теріс) байланыс- ты не жоғары, не төменгі фор- муланы таңдап алып, сол ар- қылы *у* функциясының мəнін табамыз (1.4-сурет). 2-блоктың орындалу барысында *х* айныма- лысына белгілі бір мəн беріледі де, ол мəн енгізу операторлары арқылы программаға енгізілуі тиіс. Бұдан кейін енгізілген мəннің оң немесе теріс екендігі үшінші шартты тексеру блогы арқылы айқындалады. Шарттың

“ақиқат” (иə) немесе “жалған” (жоқ) болуына байланысты 4- не 5-блоктардың бірі ғана орындалып, “таңдау” орындалады. 6-блок *х* айнымалысының жəне *у* функциясының сандық мəндерін экранға немесе қағазға басып шығарады.

1. **Циклдік алгоритмдер**. Математикада, экономикада көп- теген есептерді шығару кезеңінде бір теңдеуді пайдаланып, ондағы айнымалының өзгеруіне байланысты оны бірнеше рет қайталап есептеуге тура келетін сəттер де жиі кездеседі. Осындай қайталап орындалатын есептеу процесінің белгілі бір бөліктерін цикл деп атайды. Осы бірнеше рет қайталанатын бөлігі бар алгоритмдер тобы циклдік алгоритмдерге жатады. Циклдік алгоритмдерді пай- далану оларды кейіннен программаларда цикл операторы түрінде қысқартып жазу мүмкіндігін береді. Циклдер қайталану санының алдын ала белгілі жəне белгісіз болуына байланысты екі топқа бөлінеді. Қайталану сандары алдын ала белгілі болып келетін циклдер тобы арифметикалық цикл болып есептеледі, ал орын- далу саны белгісіз циклдер – қадамдық (итерациялық) цикл бо- лып аталады.

Практикада белгілі бір айнымалының сандық мəніне байланы- сты орындалатын арифметикалық циклдер жиі кездеседі. Мұнда арифметикалық прогрессияға ұқсас болып келетін циклдер ең қарапайым арифметикалық цикл болып табылады. Оны басқару қайталану кезеңінде прогрессияның заңына сəйкес тұрақты ша- маға өзгеріп отыратын цикл параметрінің сандық мəнімен байла- нысты болуы тиіс.

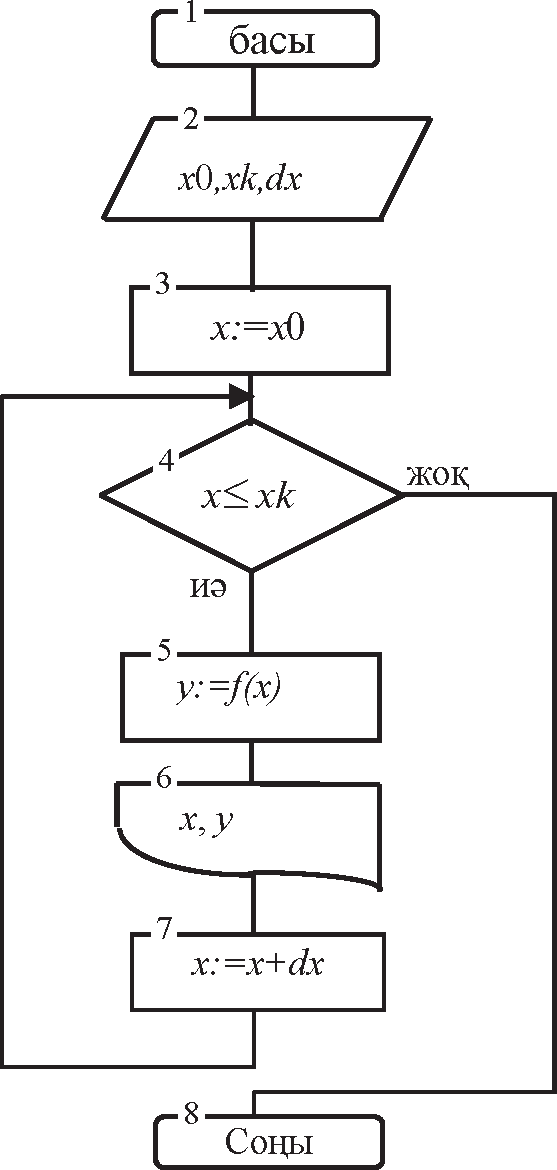
Цикл орындалуы алдында оның айнымалы аргументі – па- раметрі алғашқы мəнге ие болуы керек, сонан кейін қайталану кезеңінде цикл параметрі белгілі бір шамаға (қадамға) өзгере от- ырып, ол алдын ала берілген ең соңғы мəнге дейін жетуі қажет.

Алгоритмнің орындалу барысында цикл параметрі, мысалы, *х* өзінің ең алғашқы *х*0 мəнінен ең соңғы *хk* мəніне дейін тұрақты шамаға (*dx*) өзгеріп отырады. Осының нəтижесінде *х* мына- дай мəндерді қабылдайды: *x*0*, x*0*+dx, x*0*+*2*dx, ..., x*0*+(n-*1*)dx, xk,* мұндағы *n* – циклдің қайталану саны, ол былай анықталады:

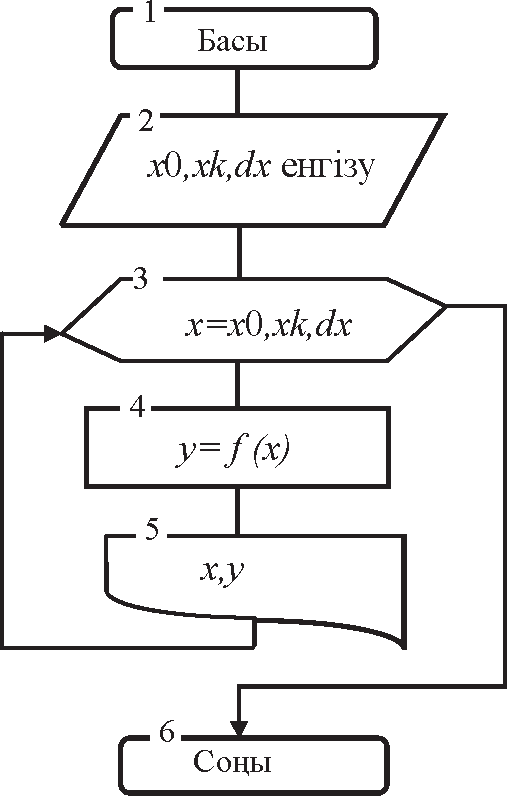
⎡ *xk* ⎤-*x*0

*n*= ⎢ ⎥ +1,

⎣⎢ *dx* ⎥⎦



**1.5-сурет.** Қарапайым циклдік алгоритм

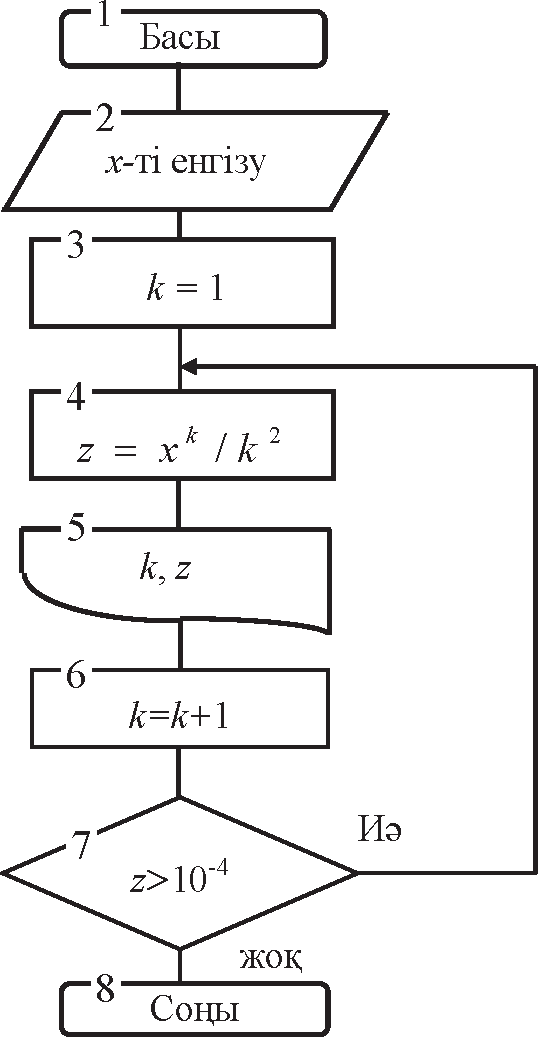


**1.6-сурет.** Модификаторлы циклдік алгоритм

мұнда [...] – өрнектің бүтін бөлігі алынатынын көрсетеді. *n* əрқашанда бүтін сан болуы тиіс, егер ол аралас сан болса, онда оның бөлшегі алынып тасталады, өйткені циклдің қайталану саны бүтін натуралдық сан болуы тиіс. Арифметикалық цикл үшін *y=f(x)* функциясының есептелу жолы алго- ритм ретінде 1.5-суретте көрсетілген. Мұндағы 3-ші, 4-ші, 7-блоктар циклді ұйымдастыру үшін қажет. Олар цикл параметрінің алғашқы мəнін, өзгеру қадамын белгілеп жəне оның ең соңғы мəніне жеткен-жетпегенін тексереді. Ал 5- жəне 6-блоктар бірнеше рет қайталанып циклдің өзін құрайды. 4-блок шартты тексеріп қайталану процесін ұйымдастырады.

Алгоритм схемасын салуды жəне

программаны жазуды жеңілдету үшін цикл алгоритмдері ықшамдалған түрде “модификатор” немесе “цикл басы” блогын пайдалану арқылы жазылады. Онда 1.5-суретте көрсетілген 3-ші, 4-ші, 7-блоктардың орнына “цикл басы” блогы орналасады. Ол алты- бұрыш тəрізді геометриялық фигура- дан тұрады жəне оның міндетті түрде екі кіру жəне екі шығу сызығы болуға тиіс. Осы блокты пайдалану арқылы жоғарыда келтірілген алгоритм 1.6-су- ретте көрсетілген түрде кескінделеді. Параметрдің алғашқы *х* мəні оның соңғы *х* мəнінен кем болса, онда оның қадамы *dx* оң сан болады. Керісінше, параметрдің алғашқы мəні оның соңғы мəнінен артық болса, онда қадам теріс сан болады.

1. **Қадамдық циклдер.** Циклді орындаудың алдында, оның қайталану саны белгісіз болған жағдайда қа- дамдық циклдер пайдаланылады. Мұн- да циклді жазу үшін тек қана “шартты тексеру” блогын қолдану қажет, ол циклді аяқтау үшін белгілі бір шартты тексереді. Қадамдық циклдердің схе- масын сызғанда модификаторды (ал- тыбұрышты) қолдана алмаймыз, себебі алдын ала циклдің неше рет қайта- ланатыны бізге белгісіз. Енді осындай циклдер жұмысына мысал келтірейік.

*xk*

*3-мысал.*

*Z*  *k* 2

функциясының

мəндерін *k* = 1, 2, 3, ... жəне *Z-*ті 0.0001- ден артық болған жағдайда есептейік, мұндағы 0  *х*  1. Бұл мысалда алдын ала цикл неше рет қайталанатынын айта алмаймыз, өйткені бізде тек *k*

**1.7-сурет.** Қадамдық цикл алгоритмі

параметрінің алғашқы мəні мен қадамы ғана белгілі. Сонымен қатар *Z* функциясының 0.0001-ден артық болуы циклді қайталау шарты болып есептеледі (*Z >* 0.001). 1.7-суретте осы есептің алгоритм схемасы көрсетілген.

* 1. **Программалау тілдері**

Алгоритмдерді ЭЕМ-ге түсінікті мəтін ретінде жазуға ар- налған қарапайым жасанды тіл программалау тілдері деп ата- лады. Əрбір ЭЕМ-нің өзінің машиналық программалау тілі бо- лады, оны командалар тілі немесе кодтар (арнайы таңбалау) тілі дейді. ЭЕМ тек өз ана тілінде, яғни машиналық тілде жазылған программаларды ғана орындай алады. Алайда, машина тілінде программа жазу өте күрделі жұмыс, өйткені ол тек екілік (он алтылық) жүйедегі кодтардан тұрады жəне əр машинада əр түрлі машиналық тіл қолданылады.

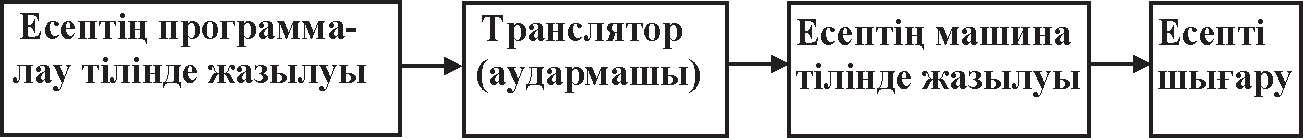
Программа жазуды жеңілдету үшін математикалық формула- ларды кеңінен қолданатын, ағылшын тілінің негізінде жасалған

алгоритмдік тілдер Бейсик, Паскаль, Фортран, СИ, т.б. кеңінен қолданылады. Алгоритмдік немесе программалау тілі – жазу ережелері қарапайым жасанды тіл. Оның машина тілдерінен айырмашылығы – табиғи ағылшын тілі негізге алынып, кең тараған математикалық таңбалармен толықтырылып жасалған. Сондықтан алгоритмдік тілдерде программа жасау адамдарға əрі жеңіл, əрі ыңғайлы болып келеді. Алгоритмдік тілдер автомат- ты түрде ЭЕМ-нің көмегімен аудармашы программалар арқылы машиналық тілге көшіріледі. Алгоритмдік тілді машина тіліне тікелей аударатын үлкен программаларды транслятор деп атай- ды. Алгоритмдік тілдерді пайдалану программалауды жеңілдете отырып, ЭЕМ-де есеп шығару процесін оңайлатады, алайда онда есеп шығару уақыты аздап көбейеді.

Алгоритмдік тілдер машинаға жəне проблемаға бағытталған болып екіге бөлінеді. Машинаға бағытталған тілдердің машина тілінен айырмашылығы, олар ЭЕМ-нің ерекшеліктерін есепке ала отырып əріптерді де пайдаланады. Қазіргі кезде машинаға бағытталған тілдерде маман программалаушылар жұмыс істейді. Оларға – автокод, макроассемблер, ассемблер тəрізді тілдер жа- тады.

Проблемаға бағытталған тілдер шығарылатын есептердің ерекшеліктерін еске ала отырып, есептің математикада жазылу тіліне жақындастырылады. Бұларға – Бейсик, Фортран, Паскаль, Си, т.с.с. тілдер жатады.

Негізінде ЭЕМ-де кез келген программалау тілінде жазылған есеп машина тіліне аударылып барып орындалады. Есептің орын- далу кезеңдерін 1.8-суретте көрсетілген схема түрінде көрсетуге болады.



**1.8-сурет.** Есепті шығару кезеңдері

Қазіргі кезде бес жүзге жуық алгоритмдік тілдер тараған. Олардың əрқайсысы белгілі бір мақсаттарда қолданылады. Мы- салы, Фортран – ғылыми-техникалық (инженерлік) есептерді шығару үшін, Паскаль – оқып үйренуде, ал Бейсик – дербес ком- пьютерлерде диалог режимінде жұмыс істеуде қолданылады.