4 ТАРМАҚТАЛУ АЛГОРИТМІ

Бұрын біз деректерді енгізу және мәндерді айнымалыларға қалай тағайындау керектігін қарастырдық. Енді бағдарлама барысында қалыптасқан жағдайға байланысты қосымшаны орындаудың әртүрлі ағындарын ұйымдастыру мәселесіне тоқталайық. Ол үшін **тармақталу** алгоритмі дегеніміз не?

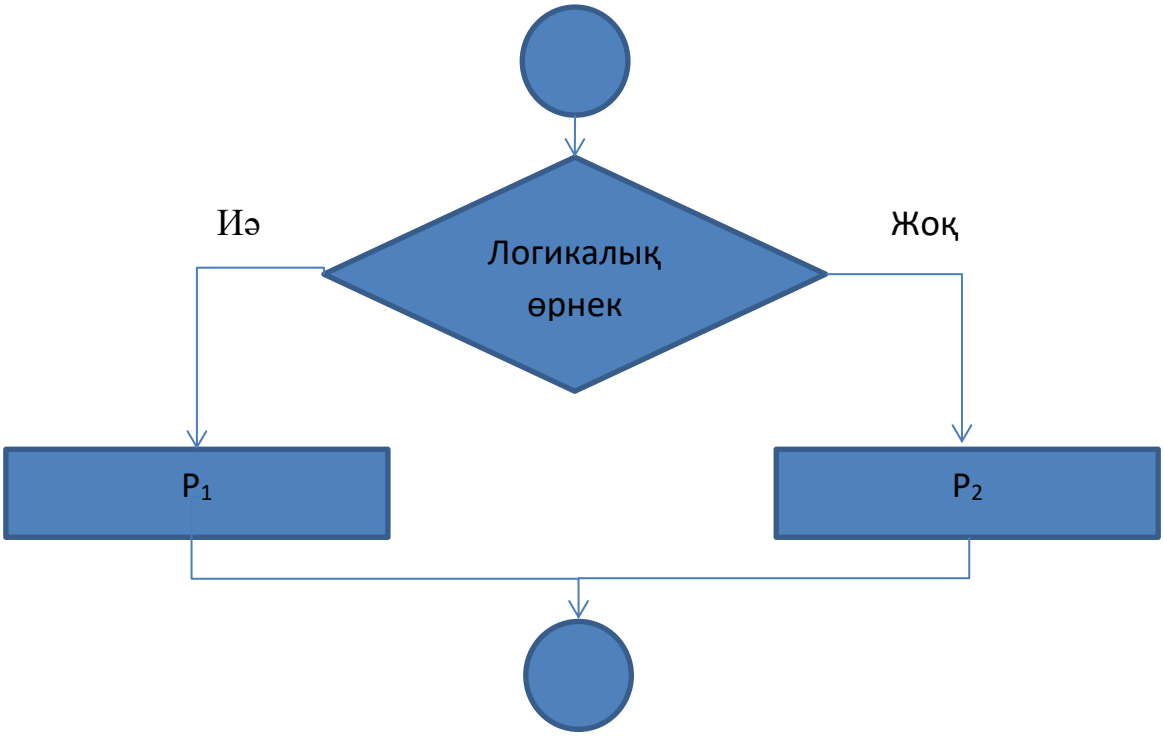
Алгоритм қадамдарының реттілігі белгілі бір шарттардың орындалуына байланысты өзгерсе, алгоритм **тармақталу** деп аталады. Шарт дегеніміз - екі мәннің біреуін қабылдай алатын логикалық өрнек: "иә" - егер шарт дұрыс болса (ақиқат), ал "жоқ" - егер шарт дұрыс болмаса (жалған).

Тармақталған алгоритмді бағдарламаларда қарапайым, қысқартылған, құрама операторлардың көмегімен, сондай-ақ көп мағыналы тармақтардың құрылымымен жүзеге асыруға болады. Төменде осы нұсқаларды толығырақ қарастырыңыз**.**

4.1 Қарапайым шартты оператор

Қарапайым шартты оператор құрылымының алгоритмінің жалпы түрі 30-шы суретте көрсетілген.

.



Сурет 30 – Шартты оператордың блок-схемасы

Қарапайым шартты оператордың синтаксисі келесідей :

if Логикалық өрнек:

P_1
else:
P_2

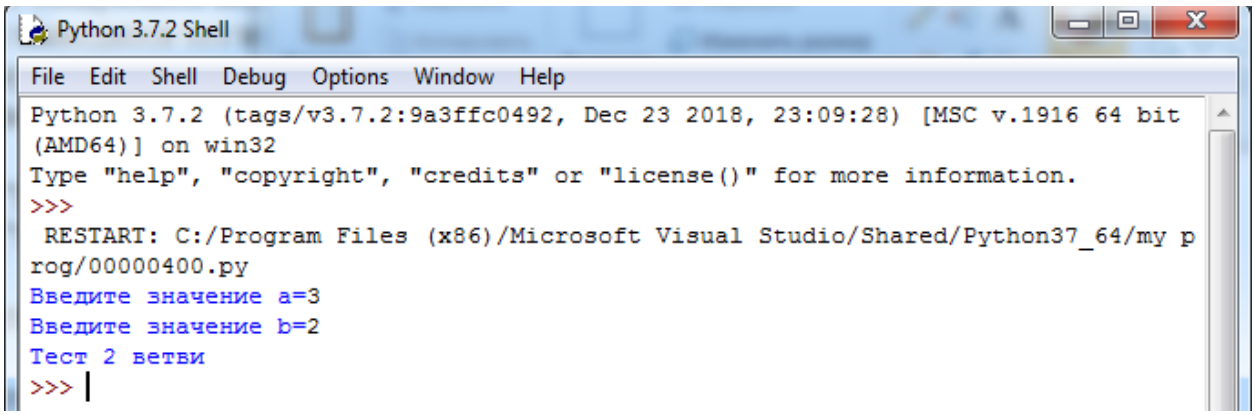
мұнда **if** (егер), **else** (әйтпесе) - сақталған сөздер, а P_1, P_2 - операторлар.

Қарапайым шартты оператор келесі алгоритм бойынша жұмыс істейді: алдымен логикалық өрнек есептеледі. Егер нәтиже **true** (ақиқат) болса, онда P1 операторы орындалады, ал P2 операторы өткізіп жіберіледі. Егер нәтиже **false**(жалған) болса онда P2 операторы орындалады, ал P1 операторы жіберілмейді.

Мысалы, келесі кодта екі айнымалыны салыстыру нәтижесіне байланысты бір немесе басқа жауап көрсетіледі. If операторы-блок операторы, сондықтан кодтағы шегіністер міндетті болып табылады. Операторлар блогына кіру қос нүкте арқылы жүзеге асырылады:

```
a=int(input("Введите значение a="))
b=int(input("Введите значение b="))
if a<b:
   print("Тест 1 ветви")
else:
   print("Тест 2 ветви")
```

31-ші суретте **a=3, b=2** мәндер үшін қарапайым шартты операторды орындау мысалының скриншоты көрсетілген

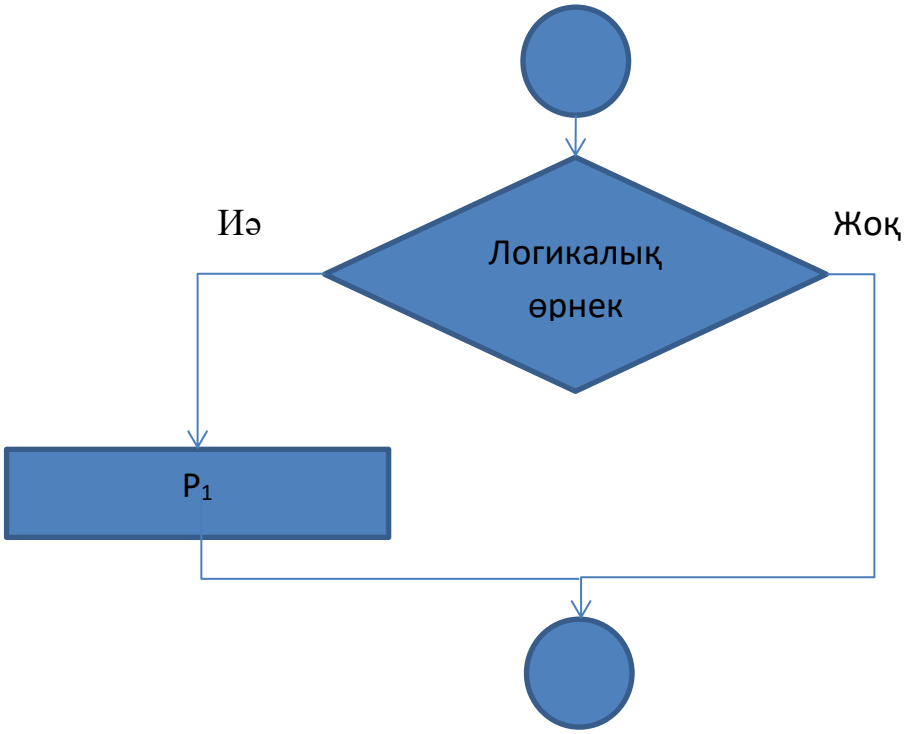


Сурет 31 – a=3, b=2 мәндері үшін қарапайым шартты операторды орындалуы

4.2 Қысқартылған шартты оператор

Егер белгілі бір әрекетті тек тексерілген **шарттың** шындығымен орындау қажет болса, онда бұл жағдайда **қысқартылған шартты оператор** қолданылады. *Қысқартылған шартты оператор* құрылымының

алгоритмінің жалпы түрі 32-ші суретте көрсетілген.



Сурет 32 – Қысқартылған шартты оператордың блок-схемасы

Қысқартылған шартты оператордың синтаксисы:

if Логикалық өрнек:
  P1

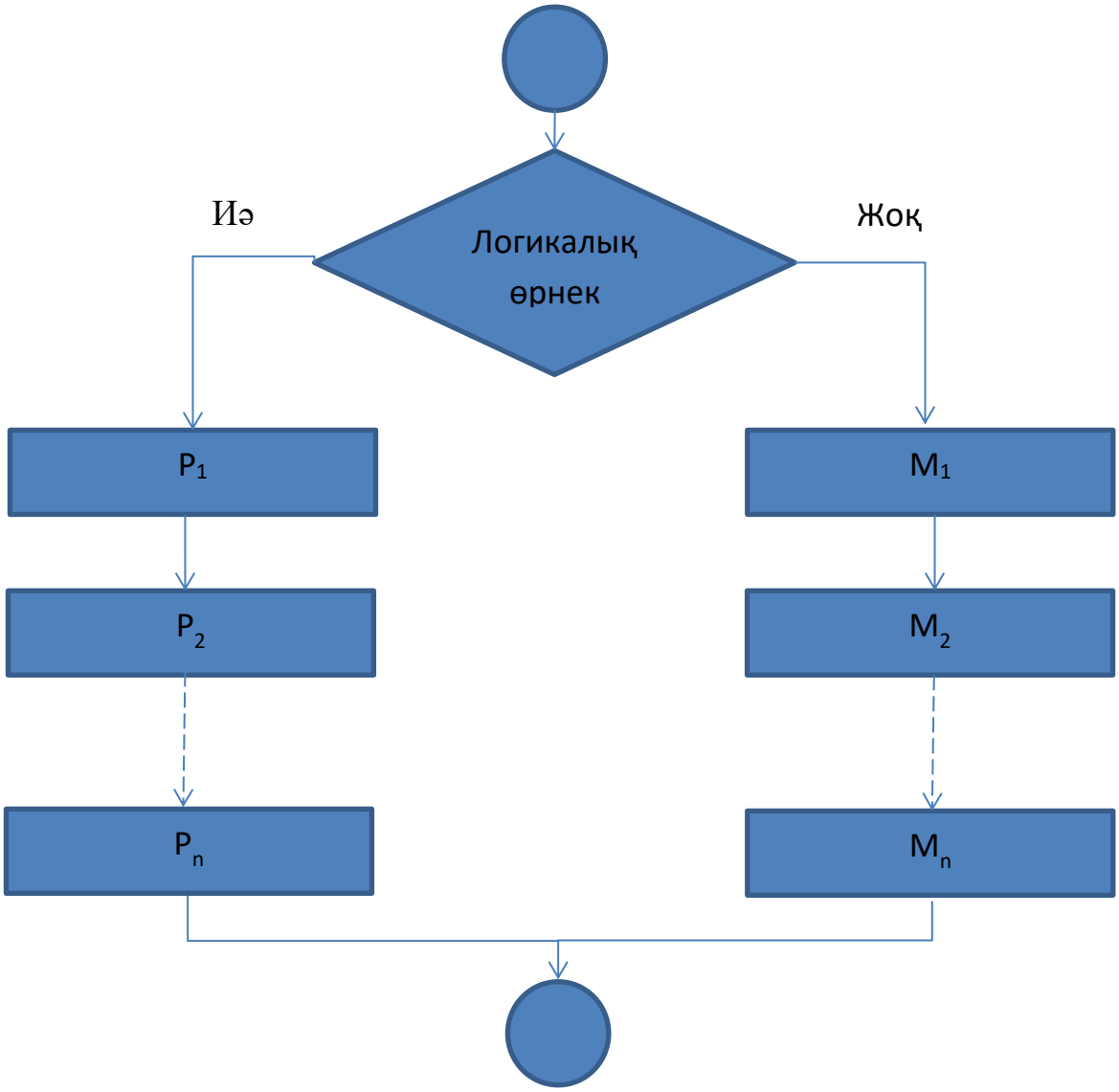
мұнда **if** (егер) – сақталған сөз, ал P_1 - оператор.

Мысалы, төмендегі листингте логикалық өрнектің ақиқатымен сәйкес хабарлама шығады " Тест 1 ветви»:

```
a=int(input("Введите значение a="))
b=int(input("Введите значение b="))
if a<b:
  print("Тест 1 ветви")
```

4.3 Құрамдас шартты оператор

Егер белгілі бір жағдайда операторлардың белгілі бір тізбегін орындау қажет болса, онда олар бір құрама операторға біріктіріледі. Құрама шартты оператор құрылысының алгоритміндегі жалпы түрі 33-ші суретте көрсетілген.



Сурет 33 – Құрамдас шартты оператордың блок-схемасы

Құрамдас шартты оператордың синтаксисы келесі түрде:

if Логикалық өрнек:
    P_1
    P_2
    .
    .
    P_n
else:
    M_1
    M_2
    .
    .
    M_n

мұндағы **if**, **else** – сақталған сөздер, ал P_1, P_2,…, P_n, M_1, M_2,…, M_n -

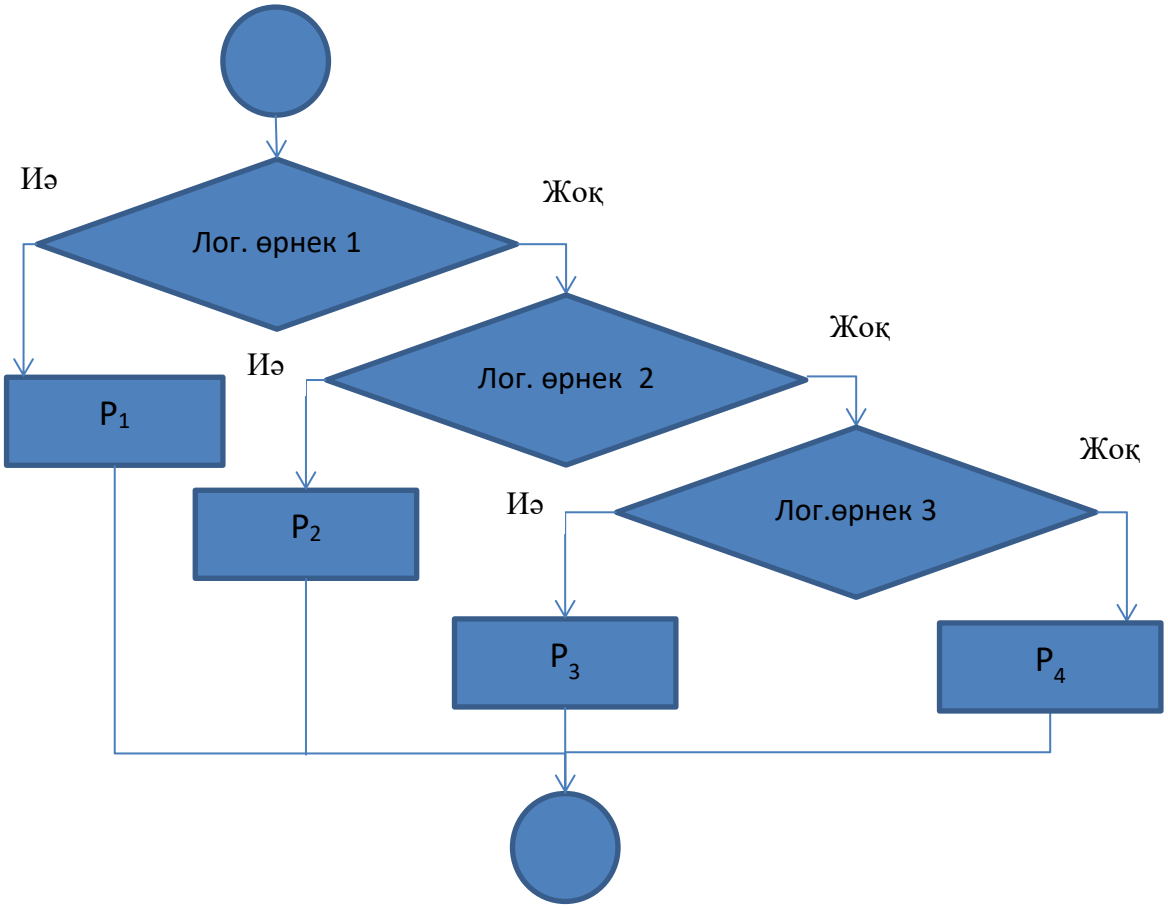
операторлар.

Мысалы, төмендегі листингте шартты оператордың әр тармағында біз екі операторды орындағымыз келеді. Содан кейін олардың әрқайсысын жолдың басынан бастап төрт бос орынға оңға жылжыту керек:

```
a=int(input("Введите значение a="))
b=int(input("Введите значение b="))
if a<b:
   a=a+b
   print("Сумма двух чисел a+b=", a)
else:
   a=a*b
   print("Произведение двух чисел a*b=", a)
```

4.4 Көп мағыналы тармақтар

Мәселені шешудің жолын екіден емес, бірнеше мүмкіндіктен таңдау өте жиі кездеседі. Бағдарламалауда бұл қадамды бірнеше шартты операторлардың көмегімен жүзеге асыруға болады. Көп мәнді тармақтарды құру алгоритміндегі жалпы көрінісі 34-ші суретте көрсетілген.



Сурет 34-Көп мағыналы бұтақтардың блок-схемасы

Көп мағыналы бұтақтардың синтаксисы келесі түрде:

if Логикалық өрнек 1:
  P_1
elif Логикалық өрнек 2:
  P_2
elif Логикалық өрнек 3:
  P_3
else:
  P_4

Мұнда **if**,**elif**, **else** – сақталған сөздер, аа P_1, P_2, P_3, P_4 - операторлар.

Бұл құрылымның жұмыс істеу алгоритмі келесіде. Егер **логикалық өрнек 1** ақиқат болса, онда осы тармақтағы оператор немесе оператор блогы орындалады, әйтпесе бұл оператор немесе блок өткізіліп жіберіледі. Егер **if** операторынан кейінгі логикалық өрнек жалған болса, онда **elif** операторынан кейінгі **логикалық өрнек 2** талданады. Егер ол шын болса, онда осы тармақтағы оператор немесе оператор блогы орындалады, әйтпесе бұл оператор немесе блок өткізіліп жіберіледі. Соңғы **else**-ден кейінгі мәлімдемелер алдыңғы барлық логикалық өрнектер жалған болған жағдайда ғана орындалады. Мұндай құрылымдағы **if** шартты операторлары **кірістірілген** деп аталады.

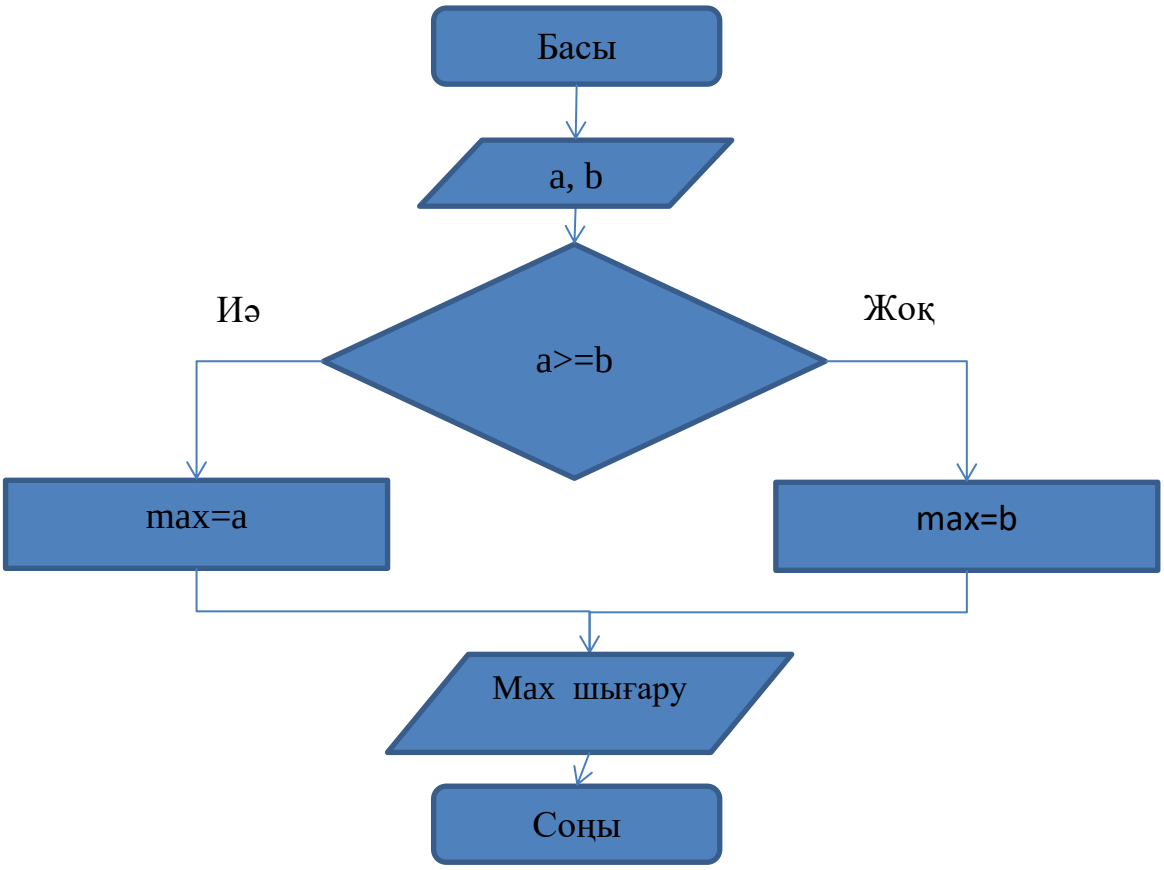
Мысалы, Листингте бағдарламаның үш тармағын тестілеу процесі көрсетілген. Пайдаланушы **a** және **b** ұяшықтарындағы бастапқы деректерді өзгерте алады және әр уақытта белгілі бір нәтиже алады.

```
a=int(input("Введите значение a="))
b=int(input("Введите значение b="))
if a<b:
   a=a+b
   print("Сумма двух чисел a+b=", a)
elif a==b:
   a=a*b
   print("Произведение двух чисел a*b=", a)
else:
   a=a-b
   print("Разность двух чисел a-b=", a)
```

4.5 Максималды және минималды элементтерді іздеу алгоритмдері

Python бағдарламалау тілін одан әрі үйренуде қажет болатын максималды және минималды мәндерді табу алгоритмдерін қарастырыңыз

Есеп 4.1. Екі санның арасындағы ең үлкенін табыңыз.
Шешімі. Есептің шешу алгоритмы 35-ші суретте көрсетілген.

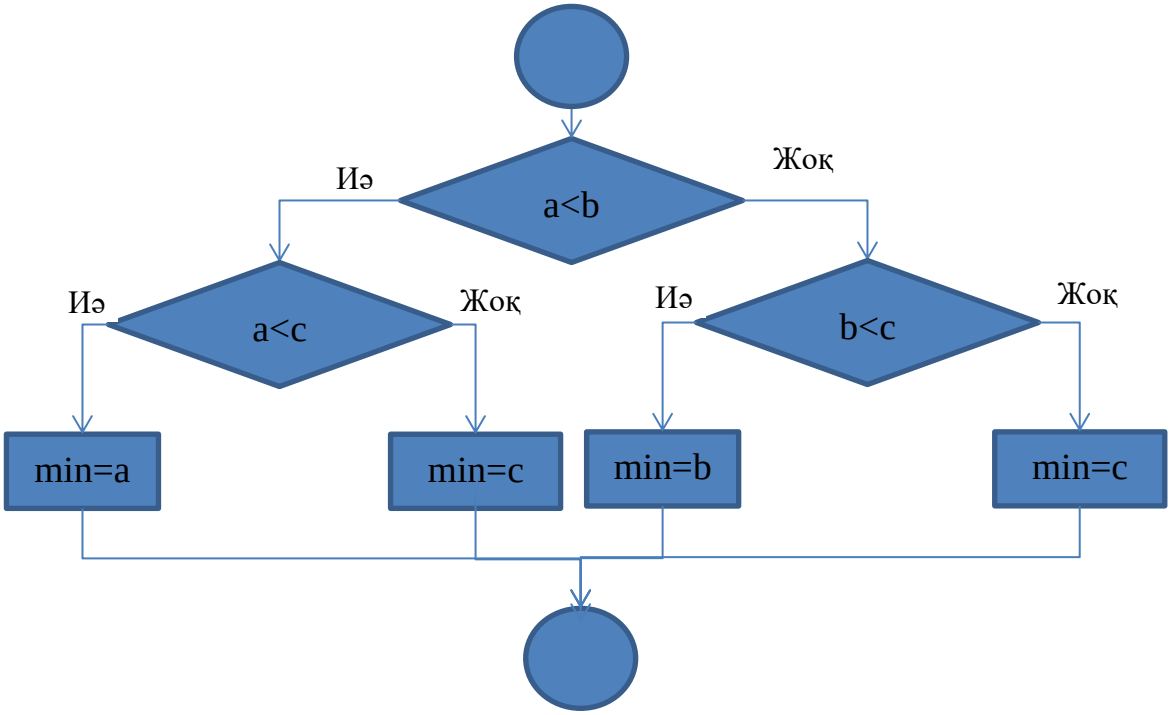


Сурет 35 – Екі санның арасындағы ең үлкен санды табу алгоритмы

Төменде листингты қараңыз:

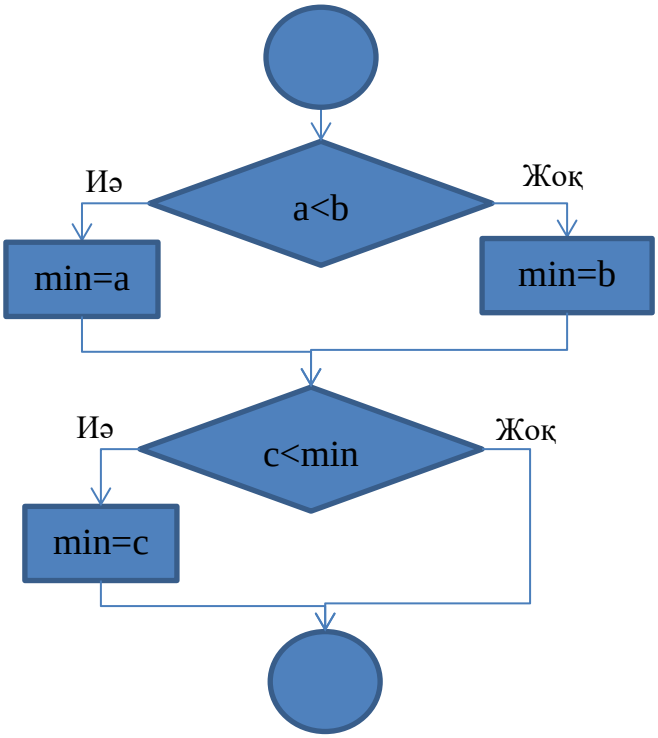
```
a=int(input("Введите значение a="))
b=int(input("Введите значение b="))
if a>=b:
   max=a
else:
   max=b
print("Екі санның үлкені max=", max)
```

Есеп 4.2. Үш санның арасында ең кішісін табу .
Шешімі. Максималды немесе минималды элементті табу алгоритмін бірнеше жолмен бағдарламалауға болады. Есепті шешу алгоритмін (бірінші әдісі) құру фрагменті 36-шы суретте көрсетілген. Көптеген сандардың ішінен экстремалды мәнді таңдау керек болған жағдайда , алгоритм өте қиын және күрделі болуы мүмкін.



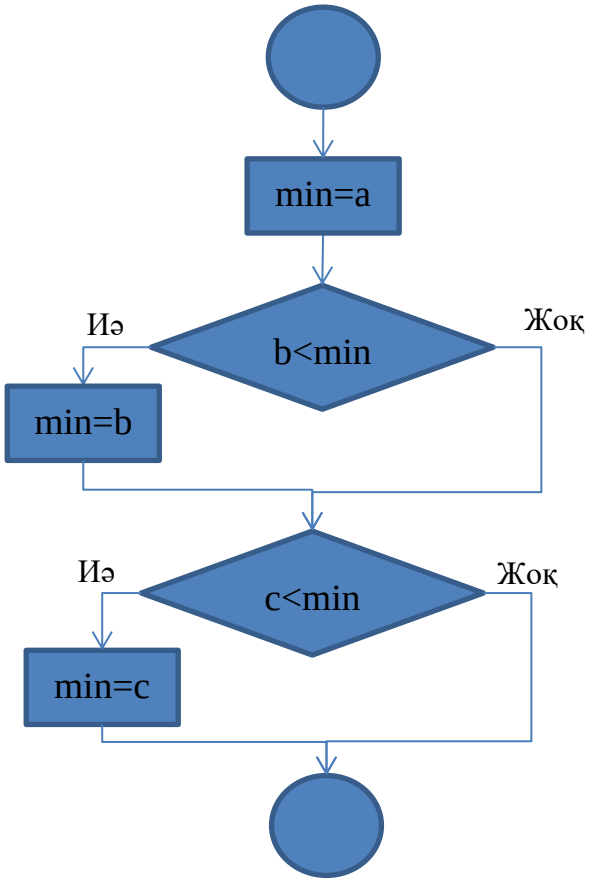
Сурет 36 – Үш санның арасынан ең кіші санды анықтау алгоритмы (бірінші әдіс)

37-ші суретте көрсетілген екінші әдіс фрагменті қарапайым және көрнекі. Екі санды бір-бірімен салыстырып, олардың ең азын анықтай отырып, әрбір келесі санды **min** ұяшығындағы санмен салыстырамыз және меншіктеу операторы ұяшықты қайта жазады.



Сурет 37 – Үш санның арасынан ең кіші санды анықтау алгоритмы (екінші әдіс)

Есепті шешу алгоритмін (үшінші әдіс) әзірлеу фрагменті 38-ші суретте көрсетілген.



Сурет 38 – Үш санның арасынан ең кіші санды анықтау алгоритмы (үшінші әдіс)

Үш санның арасынан ең кіші санды анықтау алгоритмы үшінші әдісі төмендегі листингте көрсетілген

```
a=int(input("Введите значение a="))
b=int(input("Введите значение b="))
c=int(input("Введите значение c="))
min=a
if b<min:
    min=b
if c<min:
    min=c
print("Үш санның арасындағы min=", min)
```