ШАРТТЫ ЦИКЛ ОПЕРАТОРЛАРЫ

4.1 While циклінің операторы

While циклінің операторы «шартты цикл» деп жиі аталады, өйткені цикл жұмысының аяқталуы кейбір шарттардың орындалмауымен байланысты. Циклді жазу пішімі:

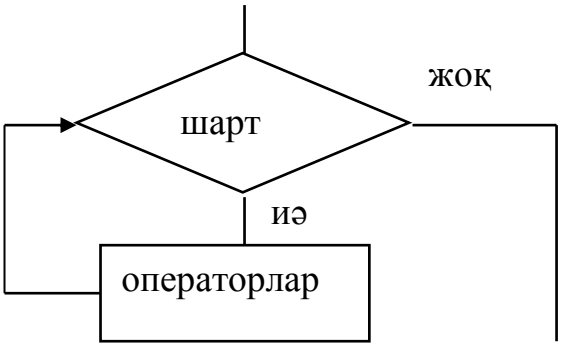
```
while (шарт)
{ операторлар; },
```

мұндағы,

`шарт` – логикалық өрнек, екі мәнді қабылдай алады: жалған немесе ақиқат.

Егер «шарт» ақиқат болса, онда while циклінің «операторлары» орындалады, әйтпесе цикл орындалмайды (аяқталады) және бағдарламаны басқару циклден кейінгі операторға көшеді.

Есептердің шешімдері бойынша алгоритмдердің құрылымдық схемасындағы `while` циклі келесі суретте көрсетілген.



4.1-суреті – `while` циклінің графикалық көрінісі

`while` циклі операторларына арналған мысалдары:

1) `while (a/x >= 0.0001) { операторлар;}`

Бұл мысалда цикл `a/x` қатынасы 0.0001 шамасынан кіші болғанға дейін орындалады.

2) `while ((x > 0) && (x <= 100)) { операторлар;}`

Бұл мысалда цикл `x` айнымалысының мәні 0-ден 100 дейінгі аралықта болғанға дейін орындалады.

4.1-есеп. Емтихан алушының столында 50 емтихан сұрақтары бар. Студент жетінші билетті тәуекелдеп алуға тырысады (алынған билет нөмері 1-ден 50 дейінгі аралықта кездейсоқ түрде құрылуы керек). Сәтсіз алынған билет емтихан алушының столына қайта қайтарылатын болса, студент билетті неше рет алады? Осыдан кейін билеттерді араластырады.

Есепте шартты циклдік процесті ұйымдастыру керек. Цикл жұмысының шарты – жетіге тең емес санды кездейсоқ түрде құру. Цикл жұмысын іске қосу алдында 1-ден 50 дейінгі аралықта орналасқан, кездейсоқ бірінші a санын құру және талпыныстар санауышына бір санын жазу керек (`c=1`). Цикл денесінде a

айнымалысы үшін кездейсоқ санның функциясы және талпыныстар санауышына (c айнымалысы) арналған инкремент операциясы болуы керек.

Цикл жұмысы аяқталғаннан кейін экран мониторына санауыш мәнін - талпыныстар санын шығаруды ұйымдастыру керек.

Бағдарлама коды мынандай:

```
using System;
namespace ConsoleCiklWhile
{
 class Program
 {
 static void Main()
 {
  int a, c;
  Random rnd = new Random();
 vvod:
  a = rnd.Next() % 50 + 1;
  c = 1;
  while (a != 7)
  {
  a = rnd.Next() % 50 + 1;
  c++;
  }
  Console.WriteLine("Билет {0}-rette tabildi", c);
  Console.WriteLine("Zhalgastiry(Y/N)?");
  char cim = (char)Console.Read();
  Console.ReadLine();
  if (cim == 'Y') goto vvod;
  Console.WriteLine("Enter pernesin basiniz");
  Console.ReadLine();
 }
 }
}
```

Бағдарлама жұмысы:

Билет 34-rette tabildi
Zhalgastiry(Y/N)?
Y
Билет 35-rette tabildi
Zhalgastiry(Y/N)?
Y
Билет 54-rette tabildi
Zhalgastiry(Y/N)?
N
Enter pernesin basiniz

`while` циклінің операторы алғыш артты цикл операторы деп аталады. Яғни, циклдің жұмыс шартын оны іске қосқанға дейін тексеру керек. Сондықтан, қарастырылған есепті шешу алгоритмінде шартты тексеруді ұйымдастыру үшін, цикл іске қосылғанға дейін бірінші сан құрылуы керек. Бұл цикл денесінің екі рет жазылуына алып келеді – цикл іске қосылғанға дейін және оның ішінде. Егер цикл денесінде екі оператор болса, онда бұл қолайсыздық онша білінбейді, бірақ көптеген операторлардың қайталануы дұрыс емес және бағдарламаның көрнекілігін бұзады (оқулықтың келесі бөлімін қараңыз).

Әдетте шартты циклдер шексіз қатарлармен берілген өрнектерді есептеу үшін қолданылады, мысалы `sin(x)` функциясын (мысал [5] алынған)

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \ldots = \sum_{i=1}^{\infty} (-1)^i \frac{x^{2i+1}}{(2i+1)!} \qquad (1)$$

Шеткі және шексіз қосындыларды есептеу алгоритмі әдетте жоғарғы деңгейдегі тілдегі бағдарламалауды оқытатын көптеген пәндерде қарастырылады. Сондықтан дәстүр бойынша `sin(x)` функциясын есептеу бағдарламасының коды мен алгоритімін қарастырайық.

`sin(x)` функциясын есептеу формуласы шексіз қосындылардан тұрады.

Математика үшін шексіздік ұғымын қолдану - әдеттегі құбылыс – бүтін және нақты сандар жиынтығы шексіз. Бағдарламалауда барлық бүтін және нақты сандардың мәндері компьютер мүмкіндігімен шектелген. Сондықтан бағдарламалауда шексіз қосындыларды есептеулер орындалмайды. Алайда қосындыны берілген дәлдікте есептеуге болады. Сонымен, шексіз қосындылар берілген дәлдіктегі қосындыларға айналдырылады.

Берілген дәлдіктегі ақырғы қосындыны есептеу алгоритмін функционалды түрде келесі әрекеттер ретімен көрсетуге болады:

бастапқы шарттарды беру;

`while(` кезекті қосылғыш модулі берілген дәлдіктен үлкен)

```
    {
қосынды = қосынды + кезекті қосылғыш;
```

жаңа мәндерді қолданып кезекті қосылғышты есептеу;

```
}
```

Бастапқы шарт мыналардан тұруға тиісті: цикл денесінің жұмысын қамтамасыз ететін қосынды және кейбір айнымалыларды инициализациялау, бірінші қосылғышты есептеу. Мысалы, кезекті қосылғыштың нөмеріне жауап беретін айнымалы.

Рекуррентті формулалар арқылы кезекті қосылғышты есептеуге кеңес беріледі. Кезекті қосылғышты есептеуді алдынғы мәнді қолданып және $a_{k+1}=f(a_k)$ рекуррентті формуласын құрып жеңілдетуге болады.

Осы тәсіл 1.1 формуласы бойынша `sin(x)` функциясын және басқа да көптеген математикалық функцияларды есептегенде қолданылады.

1.1 формуласы бойынша керекті рекуррентті қатынастардың құрылуын көрсетейік.

$$a_0 = \frac{(-1)^0 x}{1!} = x; \quad \ldots \quad a_k = \frac{(-1)^k x^{2k+1}}{(2k+1)!}; \quad a_{k+1} = \frac{(-1)^{k+1} x^{2k+3}}{(2k+3)!} \quad \ldots \qquad (2)$$

$\frac{a_{k+1}}{a_k}$ қатынасын есептеп керекті рекуррентті арақатынасын аламыз:

$$a_{k+1} = a_k \frac{-x^2}{(2k+2)(2k+3)} \qquad (3)$$

Рекуррентті арақатынасты енгізу нәтижесінде циклдің әрбір қадамында санның дәрежесін және факториалдарды табудың қажеті жоқ.

4.2-есеп. sin(x)-тің мәнін есепте, x диалог режімінде беріледі. Есептеуді екі жолмен орындау керек: стандартты sin() функциясымен және берілген t дәлдігіне сәйкес (1.1) қатар көмегімен, t мәні диалог режімінде беріледі.

Шыққан рекуррентті арақатынастарды ескере отырып келесі бағдарлама кодын жазуға болады:

```
using System;
using System;
using System.Collections.Generic;
using System.Text;
namespace ConsoleApplication1
{
 class Program
 {
 static void Main()
 {
  double sym, x, t, y;
  string buf;
  int i;
 vvod:
  Console.WriteLine("sin(x) ornegin eseptey");
  Console.Write("x= ");
  buf = Console.ReadLine();
  x = Convert.ToDouble(buf);
  Console.Write("Eseptey daldigin engizy, micali, 0,0001
");
  buf = Console.ReadLine();
  t = Convert.ToDouble(buf);
  sym = 0; y = x; i = 0;
  while (Math.Abs(y) > t)
  {
  sym = sym + y;
  y = y * (-x * x) / ((2 * i + 2) * (2 * i + 3));
  i++;
  }
  Console.WriteLine("sin(x) = {0} ", Math.Sin(x));
  Console.WriteLine("sym = {0} ", sym);
  Console.WriteLine("Zhalgastiry (Y/N)?");
```

```
  char cim = (char)Console.Read();
  Console.ReadLine();
  if (cim == 'Y') goto vvod;
  Console.WriteLine("Enter pernesin basiniz");
  Console.ReadLine();
 }
 }
}
```

Бағдарлама жұмысы:
sin(x) ornegin eseptey
x= 0,35
Eseptey laldigin engizy, micali, 0,0001 0,00001
sin(x) = 0,342897807455451
sym = 0,342897934895833
Zhalgastiry (Y/N)?
Y
sin(x) ornegin eseptey
x= 1
Eseptey laldigin engizy, micali, 0,0001 0,000001
sin(x) = 0,841470984807897
sym = 0,841471009700176
Zhalgastiry (Y/N)?

Диалог режимінде әр есептеу үшін әр түрлі дәлдіктер берілген, ол есептеу нәтижелерінің ұқсас разрядтар санын анықтайды.

Қарастырылған алгоритмді құру әдістемесін көптеген есептердің (рекуррентті арақатынастарын есептеуге болатын) ақырғы қосындысын есептеуде қолдануға болады.

Кейбір математикалық есептерде күрделі алгебралық өрнектер бірнеше қарапайым өрнектерге бөлінеді. Әрбір қарапайым өрнектек үшін өз рекуррентті арақатынасын есептеуге болады. Алгебралық өрнектерді бөліктерге бөлу (декомпозиция) және рекуррентті қатынастарды қолдану күрделі математикалық есептерді шешуге арналған алгоритмдерді құруға мүмкіндік береді.

4.2 do – while циклінің операторы

do – while циклін тексеруді цикл денесі орындалғаннан кейін жүргізу керек болған жағдайда қолдану орынды, мысалы 4.1 есебінің алгоритмін құру үшін. Циклді жазу пішімі:

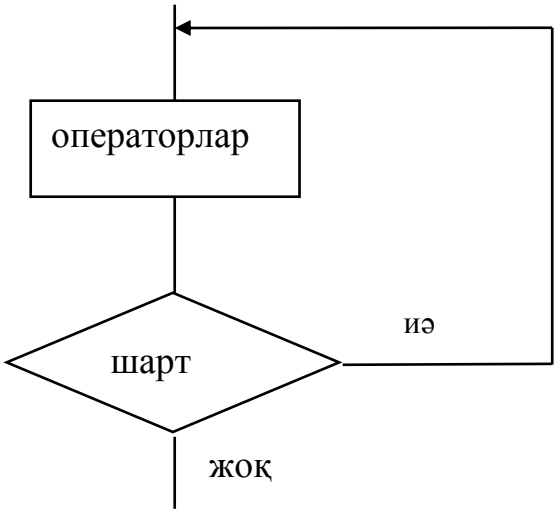
```
    do
{ операторлар; }
while (шарт),
```

мұнда,
`шарт` – логикалық өрнек, оның екі мәні болуы мүмкін: жалған немесе ақиқат.

егер «шарт» ақиқат болса, онда do – while циклінің «операторлары», әйтпесе цикл орындалмайды (аяқталады).

do – while циклі соңғы шарты цикл деп аталады - шарт цикл денесі орындалғаннан кейін тексеріледі.

Келесі суретте есептің шешімі бойынша алгоритмнің құрылымдық схемасында `do – while` циклі былай көрсетіледі.



4.2-суреті – do – while циклінің графикалық көрінісі

`do – while` циклінің ерекшелігі - цикл денесі кем дегенде бір рет орындалады.

Келесі есепті шығару үшін осы оператордың жұмысын қарастырайық.

4.3-есеп. Қатар қосындысын есептейтін бағдарлама жазу:

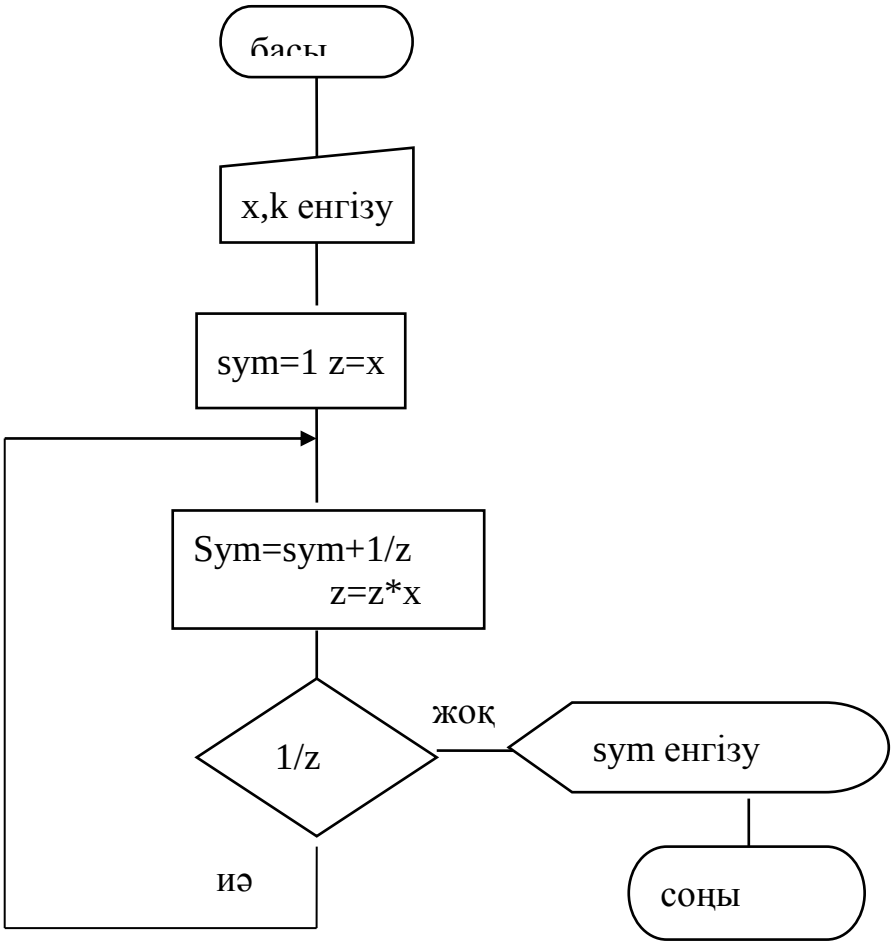
$$S = 1 + \frac{1}{x} + \frac{1}{x^2} + \frac{1}{x^3} + \ldots + \frac{1}{x^n} \qquad (4)$$

Есептеу n дәрежесіндегі `X`-ке бөлінген бір K санынан кіші болғанға дейін жалғасады, K саны ЭВМ-де диалог режімінде беріледі. `X` мәні де ЭВМ-де диалог режімінде беріледі және ол бірден үлкен болуы керек.

Күрделі математикалық өрнектердің есептеу алгоритмін құру барысында оны жеке бөліктерге бөлу (декомпозиция жасау) дұрыс, мысалы, алымы мен бөлімі. Оны орындау айтарлықтай барлық алгоритмнің құру процесін жеңілдетеді және болатын қателіктер санын азайтады.

Үлкен математикалық өрнекті бір жолмен жазуға тырыспаңыз. Оны жеке үзінділерге бөліңіз. Ол бағдарламаны дұрыстау (отладка) процесін едәуір қысқартады. Оқыту үлгісі ретінде есептің математикалық өрнегін бөліктерге бөлу (декомпозиция) процесін қарастырайық.

Мысалы, өрнектің бөлімі `X-ке тең Z (Z = X)` айнымалысымен белгіленсе, онда әрбір келесі қосылғыштың бөлімі қатардың алдыңғы бөлімін `X`-ке көбейтіндісіне тең, яғни `Z = Z * X.`



4.5– суреті – 4.3-есептің шешімі бойынша құрылымдық алгоритм

Қатардың қосылғыштарының алымы әрқашан бірге тең, сондықтан ол үшін қосымша айнымалыны қосудың қажеті жоқ.

`do while` циклін «іске қоспас» бұрын өрнекті анықтап алу керек, ол өрнек циклде шарт ретінде қолданылады. Біздің есепте қолданылатын өрнек: $\frac{1}{Z}>K$ – шарт «ақиқат» «болғанша» цикл орындала береді.

Цикл денесінде қосындыны жинау керек және жаңа `Z` бөлгішін есептеу керек.

Бағдарлама коды мынандай:

```
using System;
namespace ConsoleDoWhile
{
 class Program
 {
 static void Main()
 {
  int n = 0;
  double s = 1, z = 0, y = 0;
 vvod:
  Console.WriteLine("x>1 manin engizy");
```

```
       string buf = Console.ReadLine();
       double x = double.Parse(buf);
      Console.WriteLine("k < 1 manin engizy");
       string buf = Console.ReadLine();
       double k = double.Parse(buf);

       if (x <= 1 && k >= 0) goto vvod;
       z = x;
       s = 1; n = 0;
       do
       {
       y = 1 / z;
       s = s + y;
       z = z * x;
       n = n + 1;
       }
       while (y > k);
       Console.WriteLine("Kosindi = {0}, cikldin iske kosily
sany = {1}", s, n);
       Console.WriteLine("Enter pernesin basiniz");
       Console.ReadLine();
      }
      }
    }
```

Бағдарлама жұмысы:

```
x>1 manin engizy
2,2
k < 1 manin engizy
0,0001
Kosindi = 1,83326850772752, cikldin iske kosily sany =
12
Enter pernesin basiniz
```

4.3 Циклді пайдаланып есепті шешу мысалы

Осы бөлімде тағы бір есептің шешімін қарастырамыз, ол `for` циклімен қатар `while` циклін де қолдануды қажет етеді. Осы есепті шешкеннен кейін, басқа мысалдарда есептің толық шешімін егжей-тегжейлі қарастырмайтын боламыз, бірақ әр түрлі үзінділердің алгоритмдерін оқуды жалғастырамыз, мысалы, сұрыптау, т.б.

4.4-есеп. Үш жүз гектарлы бақшадан картоп өнімін жинау қажет болсын. Жеке тәжірибеге сүйенсек, 840 шұңқыр қазу керек. Бір шұңқырда 0-ден 5 түйнекке дейін (кездейсоқ сан) болуы мүмкін. Түйнектердің көлеміне қарай шелекте оның 70-тен 100 дейін саны болады (кездейсоқ сан). Бір қапқа төрт шелек картоп салынады. Картоп өнімін жинауға бос қаптардың қаншасын алу керек?

Есепті шешу алгоритмі шынайы алгоритмге - картопты қазу кезіндегі адамдардың әрекетіне сәйкес келеді. Біріншіден,барлық картоп (барлық 840 шұңқыр) қазып алынады және кептіру үшін бір үйіндіге жиналады. Ал содан кейін, үйіндідегі картоп біткенге дейін (`while` циклі) оны шелекке толтырады және қапқа төгеді.

Есепті шешу алгоритмін толығырақ сипаттайық. Бірінші кезеңде `for` циклін 840 циклдік операцияларға қосу керек (шұңқырлар саны бойынша), оның ішінде екі әрекетті орындаймыз: бір шұңқырда түйнектердің кездейсоқ санын құрамыз және бақшада қазып алынған түйнектердің жалпы санын жинақтаймыз.

Осылай, `for` циклінің аяғында белгілі бір айнымалыда, мысалы c, біз қазып алынған түйнектердің жалпы санын анықтаймыз.

Шелектердегі өнімді есептеу үшін «әзірше» шартты циклін (үйіндіде картоп таусылғанға дейін) қолдану керек. Цикл ішінде шелектегі түйіндердің кездейсоқ санын құру керек және осы санға түйіндердің жалпы санын азайтуды орындау мен жиналған шелектер санауышын көбейтіп отыру керек.

Осылайша шартты цикл аяқталғаннан кейін жиналған картоп салынған шелектер санын анықтаймыз. Қаптар санын есептеу үшін жиналған шелектер санын төртке бөліп, жауапты дөңгелектеу керек.

Есептелген қаптар саны есептің шешімі болады.

Бағдарлама коды мынандай:

```
using System;
namespace ConsoleFor_While
{
 class Program
 {
 static void Main()
 {
  int i, c, b, m;
  c = 0;
  //int i, j, k, a, max, min;
  //max = -100; min = 100; j = 0; k = 0;
  Random rnd = new Random();
  for (i = 1; i <= 840; i++)
  c = c + rnd.Next() % 6;
  Console.WriteLine("Barligi   {0}   kartoptin   tyinegi
zhinaldi", c);
  b = 0;
  while (c > 0)
  {
  c = c - (rnd.Next() % 31 + 70);
  b++;
  }
  if (b % 4 == 0)
  m = (int)(b / 4);
  else m = (int)(b / 4) + 1;
  Console.WriteLine("Barligi {0} shelek zhinaldi", b);
```

```
  Console.WriteLine("{0} kap kerek", m);
  Console.WriteLine("Enter pernesin basiniz");
  Console.ReadLine();
 }
 }
}
```

Бағдарлама жұмысы:
Barligi 2039 kartoptin tyinegi zhinaldi
Barligi 24 shelek zhinaldi
6 kap kerek
Enter pernesin basiniz

4.4 Өзін-өзі тексеру сұрақтары

1 Қандай жағдайда бағдарламада `while` циклін қолдану орынды?
2 `while` циклі қандай типке жатады?
3 `do {…}while` циклі қандай типке жатады?
4 "өрнектің" қандай мәнінде мына цикл тоқтатылады?
```
while ("выражение") { . . . }
```
5 "өрнектің" қандай мәнінде мына цикл тоқтатылады?
```
do { . . . } while ("выражение")?
```
6 Бағдарламаның келесі үзіндісі нені шығарады?
```
k = 5; I = 0;
while (I < k)
{
I = I + 2;
k++;
}
Console.WriteLine("I = {0} k = {1} ",I,k);?
```
7 Цикл аяқталғаннан кейін `I` айнымалысының мәні қандай?
```
I = 0;
while (I != 10)
{
I++;
. . .
}
Console.WriteLine("I = {0} ", I);?
```
8 Бағдарламаның келесі үзіндісі экранға нені шығарады?
```
I = 0;
while (I != 10)
for (I= 1;I<=9;I++) k++;
Console.WriteLine("I = {0}",I);?
```
9 Бағдарламаның келесі үзіндісі экранға нені шығарады?

```
I = 0;
while (I != 10)
{
 j = 0;
 for (I = 1; I<=9; I++) j++;
}
Console.WriteLine("j = {0} ", j);?
```

10 Бағдарламаның келесі үзіндісі экранға нені шығарады?

```
j = 1; I = 0;
while (I < 5)
{
I++;
j = j*2;
}
Console.WriteLine("j = {0} ", j);?
```