3 C# ТІЛІНІҢ КҮРДЕЛІ ОПЕРАТОРЛАРЫ

3.1 Шартты өту операторы if

Бағдарламалау тілдерінде барлық операторлар қарапайым және күрделі операторлар болып бөлінеді. Сонымен бірге күрделі оператор деп өз қызметінде басқа операторларды қолданатын операторды жатқызуға болады.

C# тілінде күрделі операторлардың бірнеше операторы болса, онда ол фигуралы жақшаға алынады.

Кең таралған күрделі операторларға шартты өту операторы және цикл операторы жатады. Әлбетте, C# тілінде басқа да күрделі операторлар бар, бірақ оқулықтың осы бөлімінде біз тек осы екі операторларды ғана қарастырамыз.

if шартты өту операторы бағдарламада есепті шешу алгоритмінің кейбір логикалық шартын тексеруді және тексерудің нәтижесіне байланысты бағдарламаның жұмысын мүмкін екі жолдың бірімен жалғастыру керек болған жағдайда қолданылады. Бағдарламаның «тармақталған» учаскесі термині де бар, оларға ауысу үшін шартты өту операторы қолданылады.

`if` операторының жазылу пішімі:

```
if ( шарт) { операторлар; } else { операторлар;}
```

Егер шарт «ақиқат» болса, онда шарттан кейін орналасқан, фигуралық жақшалар ішіндегі операторлар орындалады, әйтпесе else қызметтік сөзінен кейін орналасқан, фигуралық жақшалар ішіндегі операторлар орындалады. Мысалы,

```
if (a > b) { x = a; y = b;} else { x = b; y = a;}
```

Көрсетілген үзіндіде азаймалы ретте екі айнымалының мәндерін ретке келтіру алгоритмі қарастырылған. `a` және `b` айнымалының мәндері салыстырылады және ең жоғарғы мәні бар айнымалы `x` айнымалысына, ал ең кіші мәні бар айнымалы `y` айнымалысына меншіктеледі.

Есептің шарты бойынша екі айнымалының арасындағы ең үлкен мәнін табу және оны `x` айнымалысына меншіктеу керек болса, онда осы есептің шешімінің алгоритмін шартты өту операторы арқылы жазуға болады:

```
if (a > b) x = a; else x = b;
```

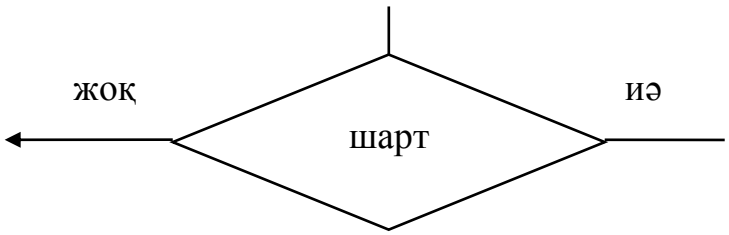
Осы жазбада операторлың іс-әрекет облысын ерекшелеу үшін фигуралық жақшалар қолданылмайды, өткені операторлың іс-әрекет облысы шарттың ақиқат және жалған мәндерінде бір ғана меншіктеу операторынан тұрады.

`if` операторының жазбасында else қызметтік сөзі болмауы мүмкін. Онда, шарт орындалмаған жағдайда бағдарлама `if` операторының сыртындағы операторды орындауға көшеді.

`Егер if` операторының іс-әрекет облысында басқа шартты өту операторлары болса, онда түсінбестікті болдырмау үшін `else` сөзі мен оның іс-әрекет облысы ең жақын сипаталған `if` операторына тиісті болады.

Есеп шішімі алгоритмінің құрылымдық схемаларын дайындағанда шарт ромб түрінде бейнеленеді, шартты жалғастырудың екі нұсқасы бар.

Шарттың графикалық көрінісі мына түрде болады:



3.1-сурет – Есеп шешімі алгоритмінің құрылымдық схемаларында шарттың графикалық көрінісі.

`if` операторының шарты алгебралық өрнектерден, салыстыру және логикалық операциялардан тұрады.

C# тілінде келесі салыстыру операциялары қолданылады:

`A == B` – егер A B-ға тең болса, нәтиже true мәніне, әйтпесе false мәніне тең болады;

`A != B` – егер A B-ға тең болмаса, нәтиже true мәніне, әйтпесе false мәніне тең болады;

`A < B` – егер A B-дан кіші болса, онда нәтиже true мәніне, әйтпесе false мәніне тең болады;

`A > B` – егер A B-дан үлкен болса, онда нәтиже true мәніне, әйтпесе false мәніне тең болады;

`A <= B` – егер A B-дан кіші немесе тең болса, онда нәтиже true мәніне, әйтпесе false мәніне тең болады;

`A >= B` – егер A B-дан үлкен немесе тең болса, онда нәтиже true мәніне, әйтпесе false мәніне тең болады;

C# тілінде логикалық операциялардың екі түрін айырады – разрядтық (поразрядные) логикалық операциялар және шартты логикалық операциялар. Разрядтық логикалық операциялар бүтін сандық типтерге қолданылады екілік (двоичном) түрде ұсынылған. Осы логикалық операцияларды қолданған кезде екілік нәтиже «0» немесе «1» болады.

Шартты логикалық операциялар әр түрлі операторлар «шарттарында» қолданылады және оның жұмысының нәтижесінің мәндері true немесе false болады.

Разрядтық логикалық операциялар:

`&` – разрядтық логикалық көбейту «ЖӘНЕ» операциясы;

`|` – разрядтық логикалық қосу «НЕМЕСЕ» операциясы;

`^` – разрядтық логикалық алып тастау «НЕМЕСЕ» операциясы.

Шартты логикалық операциялар:

`&&` – логикалық көбейту «ЖӘНЕ» операциясы;

`||` – логикалық қосу «НЕМЕСЕ» операциясы;

`!` – логикалық терістеу «ЖОҚ» операциясы.

Логикалық көбейту «ЖӘНЕ» операциясы true (ақиқат) мәнін қабылдайды, егер осы операцияның барлық көбейткіштері true мәніне тең болса.

Логикалық қосу «НЕМЕСЕ» операциясы true мәнін қабылдайды, егер осы операцияның кем дегенде бір қосылғышы true мәніне тең болса.

Өрнектің логикалық терістеу операциясы true мәнін қабылдайды, егер өрнек false мәніне тең болса. Өрнектің логикалық терістеу операциясы false мәнін қабылдайды, егер өрнек true мәніне тең болса.

Ескерту, логикалық операциялар салыстыру операцияларына қарағанда басымдығы төмен. Осыны кейбір логикалық өрнектерді жазуда ескеру керек. Мысалы, өрнегінің мәні «ақиқат» шартты өту операторын жазу керек болса, егер кейбір `x` айнымалы 0-ден үлкен, ал 10-нан кіші болса, онда бұл өрнек мына түрде жазылады:

```
if ( x > 0 && x < 10 ).
```

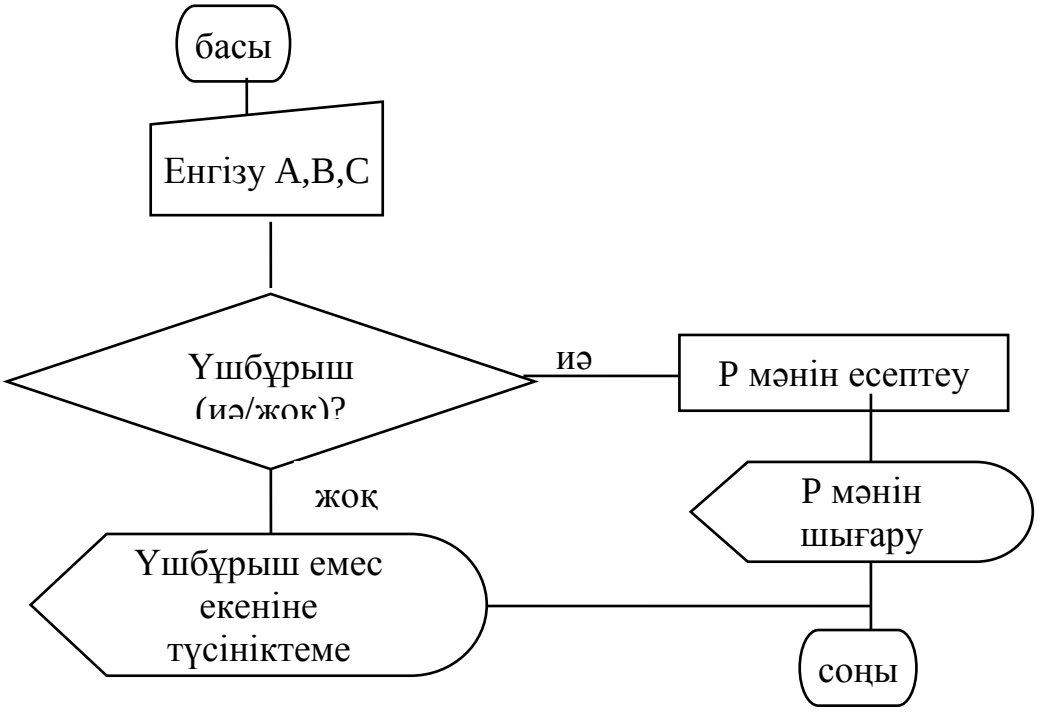
Шартты қолданатын есептің мысалын қарастырайық.

3.1 есебі. Диалог режимінде үшбұрыш жақтарын беру және оның периметрін есептеу керек. Үшбұрыш жақтарының мәндерін енгізгеннен кейін келесі тексерулерді орындау керек: үшбұрыштың барлық жақтары нөлден және кез келген екі жақтарының қосындысы үшіншісінен үлкен болуы керек. Бағдарлама жұмысына тиісті түсініктеме беру керек.

Есепті шешуге арналған құрылымдық схемасын дайындайық. Біріншіден, диалог режимінде үшбұрыш жақтарын енгізуді ұйымдастырамыз.

Одан кейін енгізілген сандармен үшбұрышты тұрғызу мүмкіндігі бойынша шарттарды тексеру керек . Егер «Иә» болса, онда үшбұрыштың периметрін есептеу және шығару керек, әйтпесе тиісті түсініктемені шығарыңыз. 3.1 есепті шешуге арналған алгоритмнің құрылымдық схемасы 3.2. суретінде көрсетілген.

Құрылымдық схемаларды бейнелегенде «төмен» және «жоғары» сызықтарының тілдері болмайды. Бірақ «солға» немесе «оңға» сызықтары болса, онда ортақ сызықтың тілі бар болуы керек.



3.2-суреті – Есепті шешуге арналған алгоритмнің құрылымдық схемасы

Есепті шешуге арналған алгоритмнің құрылымдық схемасына сәйкес бағдарлама кодын дайындаймыз.

```
using System;
namespace ConsoleApplication1
{
 class Program
 {
 static void Main()
 {
  int a, b, c;
  string buf;
 vvod:
  Console.Write("a bytin canin engiziniz ");
  buf = Console.ReadLine();
  a = Convert.ToInt32(buf);
  Console.Write("b bytin canin engiziniz ");
  buf = Console.ReadLine();
  b = Convert.ToInt32(buf);
  Console.Write("c bytin canin engiziniz ");
  buf = Console.ReadLine();
  c = Convert.ToInt32(buf);
  if (a > 0 && b > 0 && c > 0)
  if (a + b > c && a + c > b && b + c > a)
   Console.WriteLine("PERIMETR = {0}", a + b + c);
  else
  {
   Console.WriteLine("Yshbyrishtin bir kabirgasi kalgan
eki kabirganin kocindisinan ylken nemese ten"); goto vvod;
  }
  else
  {
  Console.WriteLine("Bir kabirgasi <= 0 !");
  goto vvod;
  }
  Console.WriteLine("Enter pernesin basiniz");
  Console.ReadLine();
 }
 }
}
```

Бағдарлама жұмысы:

```
a bytin canin engiziniz 1
b bytin canin engiziniz 2
c bytin canin engiziniz 3
Yshbyrishtin bir kabirgasi kalgan eki kabirganin
kocindisinan ylken nemese ten
```

```
a bytin canin engiziniz -4
b bytin canin engiziniz 5
c bytin canin engiziniz 6
Bir kabirgasi <= 0 !
a bytin canin engiziniz 7
b bytin canin engiziniz 8
c bytin canin engiziniz 9
PERIMETR = 24
Enter pernesin basiniz
```

Есепті шешуге арналған алгоритмнің «Шарты» бірнеше тексеруді жүргізеді. Тексеруді бір шартты өту операторымен немесе бірнеше шартты өту операторларымен орындауға болады, мысалы, екі.

Біздің бағдарламамызда екі `if` операторы қолданылған.

Бірінші `if` операторы үшбұрыштың барлық жақтары 0-ден үлкен болу шартын тексереді. Егер шарт орындалса, онда екінші `if` операторына көшу орындалады, әйтпесе монитор экранына "`Bir kabirgasi <= 0 !`" хабарламасы шығады және бағдарлама `goto` операторы арқылы үшбұрыш жақтарының мәндерін қайталап енгізу үшін `vvod` таңбасына (тамғасына) көшеді.

Екінші `if` операторының шартты үшбұрыш жақтарының арақатынасын тексереді. Егер шарт орындалса (бұл екі `if` операторлардың шарттары орындалуымен сайма-сай), онда монитор экранына үшбұрыш периметрінің мәні шығады және бағдарлама аяқталады. Әйтпесе, монитор экранына «`Yshbyrishtin bir kabirgasi kalgan eki kabirganin kocindisinan ylken nemese ten`" хабарламасы шығады және `goto` оператор арқылы үшбұрыш жақтарының мәндерін қайталап енгізу орындалады.

`goto` операторы C# тілінің күрделі операторы болып табылмайды, бірақ әдетте бұл шартсыз өту операторы if шартты өту операторын оқығанда қарастырылады.

Шартсыз өту операторы пішімінің жазбасы: `goto` таңба; .

Шартсыз өту операторын қолдану алдында таңбаны жазу керек (біздің мысалда `vvod` идентификаторы:), оған біз бағдарламаны бағыттаймыз. Таңба атауынан кейін қос нүкте символын қою керек. Егер бағдарламада шартсыз өту операторы кездессе, онда бағдарлама тиісті таңбадан кейін тұрған операторды орындауға көшеді.

Бір таңбаға кем дегенде бір шартсыз өту операторы сәйкес келуі тиіс.

Біздің бағдарламада есеп шарттары орындалмаған жағдайында шартсыз өту операторының көмегімен үшбұрыш жақтарының мәндерін қайталап енгізу ұйымдастырылды («дұрыс» мәндер берілгенге дейін).

Есеп шешімінің алгоритмінде ол жоқ, бірақ бағдарлама кодын жазғанда есептің шешіміне жалпы шешімді жетілдіретін кейбір үзінділерді әрдайым толықтыруға болады.

Есептің шарты бойынша бағдарламаның орындалуы түсініктемелермен толықтырылуы тиіс. Мысалда үшбұрышты құрастыру шарттары орындалмаған жағдайда түсініктемелер қолданылады.

C# бағдарламалау тілінде шартты операторы бар, ол функция рөлін атқарады және оның жұмысының нәтижесін кейбір айнымалыға меншіктеу немесе кейбір өрнекте, басқа операторда қолдану керек. Қысқаша белгіленуі – ?:.

Осы операторлың пішімі келесі түрде болады:

```
өрнек1 ? өрнек2 : өрнек3;
```

мұндағы `өрнек1` – шарт болады. Егер шарт ақиқат болса, онда `?:` операторының мәні `өрнек2` сәйкес, әйтпесе `өрнек3`. Мысалы, бағдарламаның келесі үзіндісінде осы оператор арқылы `a, b` және `c` үш айнымалысының ішінен ең жоғарғы мән анықталады:

```
x= a>b ? a : b;
x= x>c ? x : c;
```

`?:` операторын қысқартылған `if` операторы деп жиі атайды, өйткені оны кейбір есептерде шартты өту операторының орнына қолдануға болады. Алайда `if` операторы C# тілінің басқа операторларын өзіне қосуы мүмкін, ал `?:` операторының ондай мүмкіндігі жоқ.

C# тілінде бағдарлама барысының бірнеше нұсқасының ішінен бірін таңдауды ұсынатын тағы да бір `switch` – шартты операторы бар.

`switch` операторының жұмысы мен жазу пішімін массивтер тақырыбында бағдарлама менюін дайындағанда қарастырамыз.

3.2 For циклінің операторы

For циклінің операторы циклдық операциялар саны алдын-ала белгілі болған жағдайда қолданылады.

`for` операторын жазу пішімі:

```
for(өрнек1;өрнек2;өрнек3)
     { операторлар;},
```

мұнда;

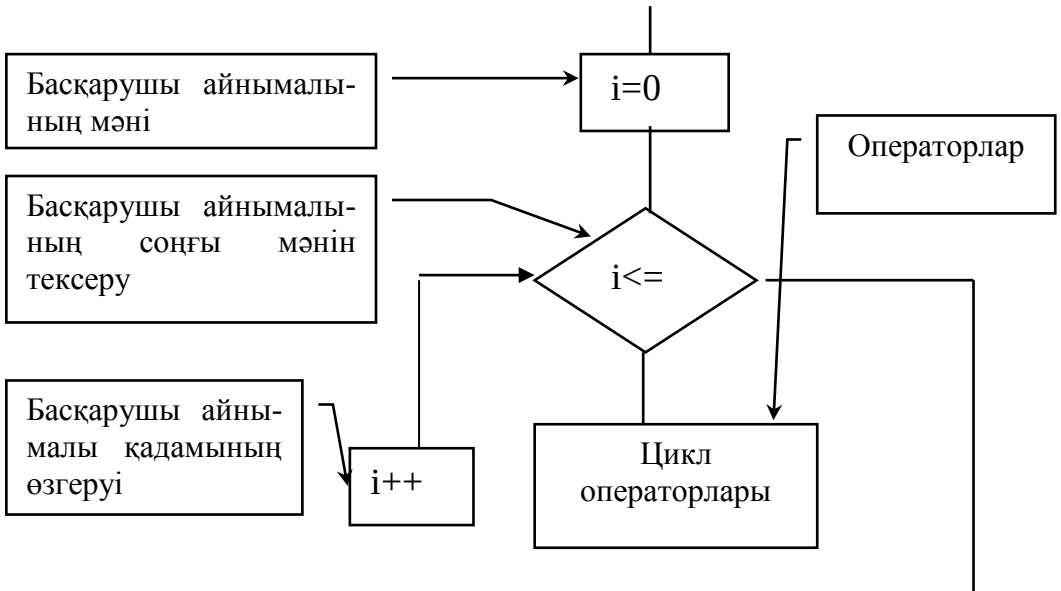
`өрнек1` – циклдің басқарушы айнымалысының бастапқы мәнін анықтайды;

`өрнек2` – циклдің басқарушы айнымалысының ақырғы мәнін анықтайды;

`өрнек3` – циклдің басқарушы айнымалысының өзгеріс қадамын анықтайды.

Мысалы: `for (i = 0; i <= 10; i++) {цикл операторлары}`

3.3 суретінде `for` циклінің графикалық көрінісі келтірілген.



3.3 суреті – For операторының графикалық көрінісі.

`for` циклін жазудың басқа мысалдарын қарастырайық:

1) `for (int i = 0; i <= 10; i++) {цикл операторлары}`

Бұл мысалда `for` циклінің операторы жазылған, онда цикл ішінде бүтін типті басқарушы айнымалы жарияланады және ол 0-ден 10-ға (қоса санағанда) дейін 1 қадамымен өзгереді.

2) `for ( i = 10; i >= 0; i--) {цикл операторлары }`

Бұл мысалда `i` – бұрынырақ жарияланған айнымалы басқарушы айнымалы ретінде қолданылады, ол 0-ден 10-ға (қоса санағанда) дейін минус 1 қадамымен өзгереді.

3) `for ( x = 0; x <= 1; x = x + 0.1) {цикл операторлары}`

Бұл мысалда басқарушы айнымалы ретінде `x` - нақты типтегі айнымалы қолданылады, ол 0-ден 1-ге дейін 0.1 қадамымен өзгереді.

3.2-есеп. минус 50-ден 50-ге дейінгі ауқымда 10 кездейсоқ бүтін сандарды басып шығару.

Есеп шешімінің алгоритмін құру жұмысын оның құрылымдық схемасымен аяқтау міндетті емес. Алгоритм жазбаша түрде, яғни есептің шешу қадамдарын толық сипаттау түрінде көрсетілуі мүмкін.

Алдымен `max` және `min` айнымалыларын жариялау керек. Оларға сәйкесінше 10 кездейсоқ сандардың ішінен ең үлкенін және ең кішісін жазамыз.

Кездейсоқ сандардың ауқымын біз білеміз, сондықтан `max` айнымалысына сандар ауқымынан кіші немесе оның ең кіші мәніне тең санды меншіктеу керек, мысалы минус 100, ал `min` айнымалысына сандар ауқымынан үлкен немесе оның ең үлкен мәніне тең санды меншіктеу керек.

Әлбетте, `max` және `min айнымалыларына` бірінші рет құрылған кездейсоқ санды меншіктеу дұрыс болады, бірақ есептің шарты бойынша олардың 10-ын құру керек, оны циклде орындаған абзал және бізге кейбір мәндерді циклге дейін меншіктеу керек. Егер меншіктеу операциясын қолданбаса, онда айнымалыларға нөл мәндері меншіктеледі, бірақ 0 саны сандар аралығының бір бөлігі, сондықтан нөлдік мәндер бағдарлама жұмысының нәтижесін бұзуы мүмкін (барлық құрылған 10 сан 0-ден үлкен немесе кіші).

Сонымен, 10 кездейсоқ санды құру керек., сондықтан `for` циклінің қадамы бірге тең, 1-ден 100-ге дейінгі аралығындағы, бүтін типті басқарушы айнымалымен бірге қолдану орынды.

Цикл денесінде кездейсоқ сандарды құруды, ал берілген аралықта оны шығаруды және осы санды бір-бірден `max` және `min` мәндерімен салыстыруды орындау керек. Егер `a` саны `max үлкен болса, онда max-ге a санын жазу керек.`

Егер `a` саны `min` кіші болса, онда `min-ге a` санын жазу керек (бірінші құрылған кездейсоқ сан `max`-нан үлкен және `min`-нен кіші болады).

10 циклдік операция аяқталғаннан кейін max және min айнымалыларының мәндерін басып шығару керек.

Бағдарламаның бастапқы коды:

```
using System;
namespace ConsoleApplication1
{
 class Program
 {
 static void Main()
 {
  int i, j, k, a, max, min;
  max = -100; min = 100; j = 0; k = 0;
  Random rnd = new Random();
  for(i=1;i<=10;i++)
  {
  a=rnd.Next() % 101 - 50;
  Console.Write(" {0}",a.ToString());
  if (a>max) {max = a; j = i;}
  if (a<min) {min = a; k = i;}
  }
  Console.WriteLine();
  Console.WriteLine("max={0} Nomer={1}",max,j);
  Console.WriteLine("min={0} Nomer ={1}",min,k);
  Console.WriteLine("Enter pernesin basiniz");
  Console.ReadLine();
 }
 }
}
```

Бағдарлама жұмысы:

```
 -33 -26 -10 -16 -44 40 -50 -35 -45 -38
max=40 Nomer=6
min=-50 Nomer=7
Enter pernesin basiniz
```

Құрылған алгоритмге біз тағы екі қосымша j және k айнымалыларын қостық, оларға құрылған сандар тізбектілігі бойынша ең үлкен және ең кіші сандарды сәйкес меншіктедік.

Кейбір бағдарламалау тілдерінде басқарушы айнымалының мәнін цикл денесінде өзгертуге үзілді-кесілді рұқсат берілмеген. C# тілінде оған рұқсат берілмеген. Мысалы, баңдарлама үзіндісі:

```
  for(i=0;i<=10;i++)
  {
  Console.Write(" {0}",i.ToString());
  if (i == 3) i = 6;
```

```
  }
 Console.WriteLine();
```

шығарады

```
0 1 2 3 7 8 9 10
```

for цикл операторының жазбасында басқарушы айнымалыны жариялауға болады. Мысалы:

```
for (int j=1;j<=10;j++) { операторлар; }
```

for циклінің ішінде j айнымалысын жариялағаннан кейін біз автоматты түрде осы айнымалының «өмірлік уақытын» анықтаймыз, ол уақыт циклдің жұмыс істеу уақытына тең. Яғни for циклі жұмысқа қосылғаннан кейін компьютердің жадынан j басқарушы айнымалының мәніне орын бөлінеді, ал цикл аяқталғаннан кейін жадының ол орны босатылады, яғни бос деп белгіленеді. Жергілікті айнымалыны жариялаудың осы әдісі бағдарламаны құруда қолданылады, оған қойылатын талаптардың бірі - компьютердің жадынан ең аз көлемді алуы керек.

for циклінің операторы бірнеше басқарушы айнымалыларын қолдануға мүмкіндік береді, мысалы:

```
for ( i=1, j=10; i<j; i + +, j - -)
```

for циклін шартты циклге айналдыруға болады, мысалы:

```
for ( i=0; (char) Console.Read() != 'n'; i++)
```

for циклінің осы жазбасы цикл жұмысының аяқталуын басқарушы айнымалымен байланыстырмай 'n' символын енгізумен байланстырады.

C# тілінде for циклінің ерекше жазу пішімі болуы мүмкін, мысалы,

```
for (i=0; i<=10) { . . . ; i++}
```

– басқарушы айнымалының өзгеру қадамы цикл тақырыбынан денесіне көшірілді;

```
for (; ;)
```

– шексіз цикл және т.б.

Қарастырылатын оқу бағдарламаларына қойылатын негізгі талап - бағдарлама жұмысы бойынша алгоритмнің көрнектілігі, ол алгоритмнің көлеміне қойылатын талапқа сәйкес бола бермейді, сондықтан for циклінің барлық ерекше жазу пішімдерін біз қарастырмаймыз.

Егер кейбір шарттарға байланысты цикл тақырыбындағы шарттардың орындалуына дейін циклдің орындалуын үзу (тоқтану) керек болса, онда break операторы қолданылады. Мысалы, for циклінің денесінде циклдің орындалу «шексіздігін» тоқтату мен бақылау керек болған жағдайларда break операторын пайдалануға болады.

C# тілінде break операторынан басқа цикл денесінің орындалуын басқаратын тағы екі оператор қарастырылған – continue және return.

continue операторы цикл денесінің соңына дейінгі барлық басқа қалған операторларды орындамай, өткізіп жібереді және циклдің келесі итерациясын іске қосады.

return - қайтару операторы күрделі оператордың немесе функцияның орындалуын аяқтайды және бағдарлама басқаруын функцияны шақыру нүктесінде немесе return операторын шақырған күрделі оператордан кейінгі келесі операторға табыстайды. Return операторы мына пішімде болады:

`return [өрнек];,`

мұндағы өрнектің және `return` функциясын шақырған оператордың типі бірдей болуы тиіс.

`Егер return` операторы C# тілінің күрделі операторын немесе `void` типті функциясын шақырса, онда өрнек болмау керек.

C# тілінде массивтер, коллекциялар деректерімен және деректер жиынын біріктіру формаларымен жұмыс істеуге арналған арнайы (`foreach`) циклі бар. Осы циклдің жұмысы мен жазылу пішімі массивтер тақырыбында қарастырылатын болады.

3.3 Өзін-өзі тексеру сұрақтары

1 `if` операторында `0<A` және `B>0` шарттарын қалай дұрыс жазуға болады, `A B`-ға тең емес?

2 `A && (! B)` өрнегі неге тең?

3 `A==B` жазбасы нені білдіреді?

4 Шартты өту операторында келесі шарттарды қандай жазба дұрыс көрсетеді:

`y =` $\sqrt{(x+z)}$ егер `z < x` және `x > 4`, әйтпесе `y =` $\sqrt{(x-z)}$?

5 C# тілінің күрделі операторларында `'{' . . . . '}'` символдары не үшін қолданылады?

6 Бағдарламада `for` циклін қашан қолдану орынды?

7 Қандай оператордың көмегімен цикл жұмысын «уақытынан бұрын» аяқтауға болады?

8 Қандай оператордың көмегімен цикл денесінің бөлігінен «өтіп кетуге» болады?

9 Бағдарламаның келесі үзіндісі нені шығарады?

```
int k=5;
int i;
for (i = k; i <= 0; i --) k = k+1;
 Console.WriteLine("i = {0} k = {1} ", i, k);
```

10 `for` циклі аяқталғаннан кейін `I` басқарушы айнымалының мәні қандай болады?

```
 int I;
  int k = 5;
  for (I = 0; I <= k; I++) k = k - 1;
  Console.WriteLine("I = {0} ", I);
```