1. Условен оператор

Програмите, които разгледахме, изпълняват операторите в реда, в който са написани. Такива програми се наричат *линейни*. Понякога обаче се налага този ред да бъде променен или да се направи избор на оператора, който трябва да се изпълни. Тук ще разгледаме един от простите методи за промяна на *потока на управление* на програмата.

1.1. Условен оператор – избор между две алтернативни условия

Синтаксис:

Кратка форма:

```
if (Условие)
   Да_оператор
```

Пълна форма:

```
 if (Условие)
    Да_оператор
else
    Не_оператор
```

Условието е **логически израз**, т.е. израз, който може да приема стойност **истина** или **лъжа.**

Семантика:

Пресмята се стойността на условието и ако то е **вярно** – изпълнява се **Да_операторът**, а ако не е вярно – **Не_операторът** (само при пълната форма). След това и при двете форми се преминава към следващия оператор.

Пример 1: Да се напише програма, която пресмята и извежда абсолютната стойност на дадено реално число *x* (*x* се въвежда от клавиатурата)

Решение:

Тъй като $|x| = \begin{cases} x, x \geq 0; \\ -x, x < 0 \end{cases}$, програмата трябва да променя знака на числото x само в случая, когато то е отрицателно. За целта ще използваме кратката форма на оператора if.

```
#include<iostream.h>
int main()
{
  int x, abs;
  abs=x;
  cout<<"Въведете стойността на x: ";
  cin>>x;
  if (x<0) abs = -x;
  cout<<"Абсолютната стойност на x е: "<<abs<<endl;
  return 0;
}
```

Операторът if проверява дали числото x е отрицателно и само в този случай променя стойността на променливата abs (съхраняваща абсолютната стойност на

числото x). Ако това условие не е изпълнено програмата продължава със следващия оператор – извеждане на абсолютната стойност на x.

Пример 2: Да се напише програма, която пресмята и извежда лицето на квадрат със страна *a*. Ако числото *a* не е положително програмата трябва да извежда съобщение, че не съществува квадрат със страна *a*.

Решение:

```
#include<iostream.h>
int main()
{
  int a,S;
  cout<<"Въведете дължината на страната на квадрата: \n";
  cin>>a;
    S=a*a;
  if (a>0)
   cout<<"Лицето на квадрата е: "<<S<<endl;
  else
    cout<<"Не съществува квадрат със страна "<<a<<endl;
  return 0;
}
```

Програмата използва пълната форма на оператора if, тъй като във всеки от двата случая (a>0 и a≤0) се извеждат различни съобщения.

Забележки:

1. Условието на ***if*** *оператора трябва да е напълно определено преди неговото използване.*

2. ***Да_оператор*** *и* ***Не_оператор*** *представляват точно един оператор*

3. Кратката форма ***if (Условие) Да_оператор*** *е еквивалентна на пълната форма* ***if (Условие) Да_оператор else ;***

Обърнете внимание, че в Пример 2 се прави излишно пресмятане на лицето на квадрата в случая, когато дължината на страната е отрицателна. По-приемливо е операторът S=a*a; да се изпълнява само в случая, когато a>0. В такъв случай обаче, ако условието на if оператора има стойност **истина** ще трябва да се изпълнят два оператора – присвояване на стойност на променливата S и извеждане на резултата, което не е допустимо. Проблеми от такъв характер се избягват чрез организирането на *съставен оператор.*

1.2. Съставен оператор. Блок. Област на видимост на променлива

Съставният оператор представлява група оператори, заключени във фигурните скоби “{” и “}”. Използва се там, където е допустим само един оператор.

Съставен оператор

Синтаксис:

```
{
        Оператор_1
        Оператор_2
        .....................
        Оператор_N
}
```

Чрез използването на съставен оператор програмата от Пример 2 може да се модифицира по следния начин:

```
#include<iostream.h>
int main()
{
  int a,S;
  cout<<"Въведете дължината на страната на квадрата: \n";
  cin>>a;
    if (a>0)
      {
       cout<<"Лицето на квадрата е: "<<S<<endl;
       S=a*a;
       }
  else
    cout<<"Не съществува квадрат със страна "<<a<<endl;
  return 0;
}
```

В C++ е допустимо обявяването на променлива да става на произволно място в програмата, в това число и във вътрешността на съставен оператор. В такъв случай съставния оператор се нарича **блок**, а обявените в него променливи - **локални променливи**. Тези променливи са достъпни само от вътрешността на блока и от никое друго място в програмата. Затова блокът представлява **област на видимост** за съответните променливи. При завършване на изпълнението на блока, паметта, резервирана за локалните променливи се освобождава. По тази причина многократното дефиниране на променливи с едно и също име в различни части на програмата е абсолютно допустимо, стига техните области на видимост да не съвпадат. За компилатора това са различни променливи.

Ако блокът е вмъкнат (**вложен**) в друг блок, в него са достъпни всички променливи, декларирани във външния блок.

1.3. Логически израз

Както видяхме, условието на if оператора трябва да бъде логически израз.

Под *логически израз* ще разбираме произволно условие, което може да провери за истинност , т.е. израз, който има стойност истина или лъжа.

Логически израз в езика C++ може да се конструира чрез основните **логически операции** – сравнение, логическо отрицание, логическо умножение (конюнкция), логическо събиране (дизюнкция).

В следващата таблица са дадени операторите за сравнение в езика C++ и техните математически еквиваленти.

оператор	= =	!=	<	<=	>	>=
мат. еквивалент	=	≠	<	≤	>	≥

Обърнете внимание, че някои оператори се представят чрез двойка символи.

Останалите логически операции се представят по следния начин:

Логически операции

“И” (конюнкция)

Синтаксис:

(Условие_1) && (Условие_2)

Семантика:

Операцията връща стойност **истина**, само когато и двата операнда имат стойност истина. В противен случай операцията връща стойност **лъжа**.

“ИЛИ” (дизюнкция)

Синтаксис:

(Условие_1) || (Условие_2)

Семантика:

Операцията връща стойност **истина**, когато поне един от двата операнда има стойност истина. В противен случай операцията връща стойност **лъжа**.

“НЕ” (логическо отрицание)

Синтаксис:

!(Условие)

Семантика:

Операцията е унарна и връща стойност **истина**, когато операндът има стойност **лъжа**, и лъжа – в противен случай.

Примери:

1. 7 = = 5 → лъжа
2. 7 != 5 → истина
3. 8 >= 8 → истина
4. 8 < 8 → лъжа
5. (3 < 5) && (20 > 0) → истина
6. (a < b) && (b <= a) → лъжа
7. (a < b) || (b <= a) → истина
8. (7 % 3 =0) || (7 / 3 = 0) → лъжа
9. !(3 > 10) → истина
10. !((a < b) || (b <= a)) → лъжа

За да се определи правилно стойността на един сложен логически израз (състоящ се от няколко операции), трябва да се знае реда, в който се изпълняват тези операции. Този ред може да бъде променен чрез кръглите скоби “(” и “)”.

Приоритет на операциите

(1) изразите в скоби: ( );
(2) унарни операции: +, -, ! ;
(3) умножение, деление: * , /, %, &&;
(4) събиране, изваждане: +, - , || ;
(5) сравнение: <, <=, >, >=, = =, !=

Внимание!
Замяна на операция “= =” с “=”

При такава замяна компилаторът няма да издаде съобщение за грешка. Напротив – ще изпълни операторът. Резултат обаче едва ли ще бъде очакваният от програмиста.

Пример:

```
int a, x = -5;
if (x = 10) a = -1;
else a = 1;
```

След изпълнението на този фрагмент променливата a ще има стойност -1, а не 1, както се очаква. Това се получава, защото логическата константа **лъжа** се представя в паметта на компютъра чрез числото **0**, а константата **истина** – чрез произволно друго число. Т.е. когато компилаторът проверява дали условието на if оператора е изпълнено (вярно), той всъщност проверява дали стойността му е различна от 0. В нашия случай условието представлява оператор за присвояване.След изпълнението му променливата x има стойност 10, която става стойност и на условието на оператора if. За компилатора това означава, че условието е изпълнено и той преминава към оператора a = -1;, а не към алтернативния оператор.

Внимание!
Използване на двойно неравенство

Пример:

```
if (0 < x < 1)
    cout<< “Не бива да се пише така \n”;
```

При използване на такъв запис програмата отново ще бъде скомпилирана и ще работи, но няма да дава верен резултат. Двойните неравенства се записват като две единични, обединени с операцията “И”. Напр. (0 < x) && (x < 1).

1.4. Вложени условни оператори

Да разгледаме следния

Пример: Да се напише програма, която намира решенията на линейното неравенство $ax+b>0$. Коефициентите a и b се въвеждат от клавиатурата.

Решение:

Тъй като решенията на неравенството зависят от стойностите на коефициентите a и b, трябва да се разгледат общо четири случая:

$$x:\begin{cases} a=0 \begin{cases} b>0, \text{всяко число е решение} \\ b\geq 0, \text{неравенството няма решение} \end{cases} \\ \text{a}\neq 0 \begin{cases} a>0, x>-\dfrac{b}{a} \\ a<0, x<-\dfrac{b}{a} \end{cases} \end{cases}$$

За да се реши задачата трябва да се разгледат две алтернативи – a=0 и a≠0, и за всяка от тях – още по две алтернативи. Това може да постигне като на местата на Да_оператора и Не_оператора в условния оператор се поставят нови два условни оператора. Такова вмъкване на един условен оператор в друг условен оператор се нарича влагане на условни оператори.

```
#include<iostream.h>

int main()
{
  double a, b;

  cout<<"Въведете коефициентите a и b \n";
  cin>>a>>b;

  if (a==0)
     if (b>0)
     cout<<"Всяко число е решение на неравенството "<<a<<"x+("<<b<<")>0\n";
     else
     cout<<"Неравенството "<<a<<"x+("<<b<<")>0 няма решения\n";
```

```
    else
       if (a>0)
       cout<<"Решения на неравенството "
            <<a<<"x+("<<b<<")>0 са всички числа, по-големи от "
            <<-b/a<<endl;
       else
       cout<<"Решения на неравенството "
            <<a<<"x+("<<b<<")>0 са всички числа, по-малки от "
            <<-b/a<<endl;

    return 0;
  }
```

Обърнете внимание на начина, по който е записана програмата: всеки else е записан точно под съответния if.

Понякога при влагане на условни оператори не е ясно кой else чия алтернатива представя. Например във фрагмента

```
if (age<18) if (age<16) cout<<"Научи си уроците \n"; else cout<<"Можеш да излезеш с приятели \n ";
```

не е ясно кога ще се изведе съобщението "Можеш да излезеш с приятели" – дали когато стойността на променливата age е между 16 и 18, или когато е по-голяма от 18. Правилото гласи, че

всяко else се свързва с последния преди него несвързан if

Т.е. в случая текстът "Можеш да излезеш с приятели" ще се изведе, когато 16≤age≤18. Препоръчва се в такива случаи за по-голяма яснота да се използва съставен оператор. С помощта на същия оператор може да се промени реда на свързване на if и else.
Например, ако искаме изречението "Можеш да излезеш с приятели" да се извежда, когато age>18 горния фрагмент трябва да се редактира по следния начин:

```
if (age<18) {if (age<16) cout<<"Научи си уроците \n";} else cout<<"Можеш да излезеш с приятели \n ";
```

Сега, ако условието age<18 е вярно ще се изпълни съставния оператор {if (age<16) cout<<"Научи си уроците \n";}, а ако то не е вярно – операторът cout<<"Можеш да излезеш с приятели \n ";

Препоръка: **При работа с вложени условни оператори, те да се влагат в съставен оператор**

ВЪПРОСИ И ЗАДАЧИ:

1. Каква ще бъде стойността на променливата y след изпълнение на следния фрагмент:

```
int x=10, y;
if (x>0) y=1;
else
  if (x<0) y=-1;
  else
    if (x = =10) y=10;
    else y=0;
```

Отговор: Тъй като x>10 y ще приеме стойност 1.

2. Да се напише фрагмент от програма, който извежда изразът “Честито!”, ако стойността на променливата exam е не по-малка от 4.50, и съобщава “Ще се видим отново през поправителната сесия”, в противен случай.

3. Да се редактира фрагмента от Зад.2, така че да

а) извежда “Честито!”, ако стойността на променливата exam е не по-малка от 4.50, а стойността на променливата programs_num – по-голяма от 20;

б) извежда “Опитай отново следващата седмица”, ако стойността на променливата exam е не по-малка от 4.50, а стойността на променливата programs_num – не по-голяма от 20;

в) извежда “Ще се видим отново през поправителната сесия”, ако стойността на променливата exam е по-малка от 4.50

4. Да се напише програма. която проверява дали числата a, b и c могат да бъдат дължини на страни на триъгълник.

Упътване:

Числата a, b и c числа могат да бъдат дължини на страни на триъгълник, ако е изпълнена системата неравенства:

$$\begin{cases} a+b>c \\ a+c>b \\ b+c>a \end{cases}$$

5. Да се напише програма, която намира и извежда по-малкото от числата *a* и *b*.

6. Да се напише програма, която намира и извежда най-голямото от числата a, b и c.

Решение:

```
#include<iostream.h>

int main()
{
  int a, b, c, max;

  cout<<"Въведете числата a,b и c\n";
  cin>>a>>b>>c;

  max=a;
  if (b>max)max=b;
  if (c>max)max=c;
  cout<<max<<endl;
  return 0;
}
```

7. Да се напише програма, която извежда числата a, b и c, сортирани във възходящ ред.

Решение:

```
#include<iostream.h>

int main()
{
  int a, b, c, max;
  int x;
  cout<<"Въведете числата a,b и c\n";
  cin>>a>>b>>c;
```

```
  if (b<a)
  {
    x=a; a=b; b=x;              //разменят се стойностите на a и b
  }
  if (c<a)
  {
    x=a; a=c; c=x;
  }
  if (c<b)
  {
    x=b; b=c; c=x;
  }
  cout<<a<<" "<<b<<" "<<c<<endl;

  return 0;
}
```

8. Да се напише програма, която проверява дали точка P(x,y) е вътрешна за квадрат с дължина на страната *a* и център – началото на координатната система.

Решение:

```
#include<iostream.h>

int main()
{
  double x, y, a;
  cout<<"Въведете координатите на т.P\n";
  cin>>x>>y;
  cout<<"Въведете дължината на страната на квадрата\n";
  cin>>a;

  a=a/2;
  if (a>0)
    if ((x>=-a)&&(x<=a)&&(y>=-a)&&(y<=a))
     cout<<"Точка P е вътрешна за квадрата\n";
    else
     cout<<"Точка P не е вътрешна за квадрата\n";
  else
    cout<<"Не съществува квадрат със страна "<<2*a<<endl;


  return 0;
}
```

9. Да се напише програма, която проверява дали т.P(x,y) е вътрешна за всяка от следните фигури:

фиг. 3

Упътване:

Използвайте следните условия за if-операторите:

а) x*x + y*y <= 25

б) (x*x + y*y <= 64) && (x*x + y*y >= 25)

в) (x >= -5) && (x <= 5) && (y >= -2) && (y <= 2)

г) (x>=0) && (y >= 0) && (y<= -0.5*x+2)

Изразът в последните скоби се определя след като се реши системата $\begin{vmatrix} 2 = 0 + n \\ 0 = 4k + 2 \end{vmatrix}$ и се прецени в коя полуравнина спрямо правата с уравнение $y=kx+n$ се намира защрихованата област

д) (y>=0) && (y<=5) && (x>=-5) && (x<=5) || (y<0) && (x*x + y*y <= 25)
е) (y>=0) && (y<= x +5) && (y<= -x +5) || (y<0) && (x*x + y*y <=25)

10. На кой от следните оператори е еквивалентен условния оператор

```
if (a<b) if (a>c) a=b-c;
else a=b+c;
```

```
а) if (a<b)
     {
       if (a>c) a=b-c;
       else a=b+c;
      }
```

```
б) if (a<b)
     {
       if (a>c) a=b-c;
      }
     else a=b+c;
```

```
в) if ((a<b)&&(a>c)) a=b-c;
   else a= b+c;
```

Отговор: а)

11. Съставете програма за пресмятане на стойността на функцията

$$f(x) = \begin{cases} x^2 - 9, & x \le 3; \\ x - 3, & 3 < x \le 5; \\ x^2 - 25, & x > 5; \end{cases}$$

12. Как трябва да се възприемат операторите

а) if (a = = b) if (a = = c) x = a; else if (b > c) x = c; else x = a;
б) if (x != 1) if (x > 0) z = 1 – x;
в) if (x – y > 0) ; else if (x –y <0) x = y;

Напишете ги така, че да се разбира кой оператор къде се влага.

Решение:

а)

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

б)

```
if (x != 1)
   if (x > 0)
      z = 1 – x ;
```

в)

```
if (x – y >0)
 {
 }
 else
    if (x – y <0)
       x = y;
```

13. Да се напише програма, която намира корените на уравнението:

а) $ax + b = 0$

б) $ax^2 + bx + c = 0$

Решение:

б)

```
#include<iostream.h>
#include<math.h>

int main()
{
  double a,b,c;
  double D,x1,x2;

  cout<<"Въведете коефициентите a, b и c\n";
  cin>>a>>b>>c;

  if (a==0)
  {
    if (b==0)
    {
      if (c==0)
      {
        cout<<"Всяко число е решение\n";
      }
      else
      {
        cout<<"Уравнението няма решение\n";
      }
    }
    else
    {
      x1=-c/b;
      cout<<"Решение на уравнението е x="<<x1<<endl;
    }
  }
  else
  {
    D=b*b-4*a*c;
    if (D>=0)
    {
      x1=(-b+sqrt(D))/(2*a);
      x2=(-b-sqrt(D))/(2*a);
      cout<<"Решения на уравнението са x1="<<x1<<" и x2="<<x2<<endl;
    }
    else
    {
      cout<<"Уравнението няма реални решения\n";
    }
  }

  return 0;
}
```

Забележка:

Програмата използва функцията sqrt(double x), която връща квадратен корен от x. Тъй като функцията е описана в библиотека със заглавен файл math.h, тази библиотека е включена чрез директивата #include<math.h>.