Глава 4

Основни структури за управление на изчислителния процес

В тази глава ще разгледаме управляващите оператори в езика. Ще ги наричаме само оператори.

4.1. Оператор за присвояване

Това е един от най-важните оператори на езика. Вече многократно го използвахме, а също в глава 2 описахме неговите синтаксис и семантика. В тази глава ще го разгледаме по-подробно. Ще напомним неговите синтаксис и семантика.

Синтаксис

<променлива> = <израз>;

- където <променлива> е идентификатор, дефиниран вече като променлива,
- <израз> е израз от тип, съвместим с типа на <променлива>.

Семантика

Намира се стойността на <израз>. Ако тя е от тип, различен от типа на <променлива>, конвертира се ако е възможно до него и се записва в именуваната с <променлива> памет.

• Ако <променлива> е от тип <bool>, <израз> може да бъде от тип bool или от кой да е числов тип.

• Ако <променлива> е от тип double, всички числови типове, а също типът bool, могат да са типове на <израз>.

• Ако <променлива> е от тип float, типовете float, short, unsiged short и bool, могат да са типове на <израз>. Ако <израз> е от тип int, unsigned int или double, присвояването може да се извърши със загуба на точност. Компилаторът предупреждава за това.

• Ако <променлива> е от тип int, типовете int, long int, short int и bool, могат да са типове на <израз>. В този случай ако <израз> е от тип double или float, дробната част на стойността на <израз> ще бъде отрязана и ако полученото цяло е извън множеството от стойности на типа int, ще се получи случаен резултат. Компилаторът издава предупреждение за това.

• Ако <променлива> е от тип short int, типовете short int и bool, могат да са типове на <израз>. В противен случай се извършват преобразувания, които водят до загуба на точност или даже до случайни резултати. Много компилатори не предупреждават за това.

Ще отбележим, че в рамките на една функция не са възможни две дефиниции на една и съща променлива, но на една и съща променлива може да ѝ бъдат присвоявани многократно различни стойности.

Пример: Не са допустими

```
…
double a = 1.5;
…
double a = a + 5.1;
…
```

но са допустими присвояванията:

```
…
double a = 1.5;
```

```
…
a = a + 34.5;
…
a = 0.5 + sin(a);
…
```

В езика C++ са въведени някои съкратени форми на оператора за присвояване. Например, заради честото използване на оператора:

```
a = a + 1;
```

той съкратено се означава с

```
a++;
```

Въведено е също и съкращението a-- на оператора a = a-1;

В същност ++ и -- са реализирани като постфиксни унарни оператори увеличаващи съответно намаляващи аргумента си с 1. Приоритетът им е един и същ с този на унарните оператори +, - и !.

Забележка: От оператора ++, за добавяне на 1, идва името на езика C++ - вариант на езика C, към който са добавени много подобрения и нови черти.

Допълнение: Операторът за присвояване = е претоварен и с функцията на дясноасоциативна инфиксна аритметично-логическа операция с приоритет по-нисък от този на дизюнкцията ||. Това позволява на оператора за присвояване

```
X = y;
```

където x е променлива, а y – израз, да се гледа като на израз от тип – типа на x и стойност – стойността на y, ако е от типа на x или стойността на y, но преобразувана до типа на x.

Пример: Програмата

```
#include <iostream.h>
int main()
{int a;
 double  b;
 b = 3.2342;
 cout << (a = b) << "\n";
 return 0;
}
```

е допустима. Резултът от изпълнението ѝ е 3, като компилаторът издава предупреждение за загуба на информация при преобразуването от тип

double в тип int. Изразът a = b е от тип int и има стойност 3. Ограждането му в скоби е необходимо заради по-ниския приоритет на = от този на <<.

Допълнение: Допустим е операторът:

X = y = 5;

Тъй като = е дясноасоциативен, отначало променливата y се свързва със 5, което е стойността на израза y = 5. След това x се свързва с 5, което е стойността на целия израз.

Някои компилатори издават предупреждение при тази употреба на оператора =. Затова не препоръчваме да се използва = като аритметичен оператор.

Задачи върху оператора за присвояване

Задача 12. Нека са дадени дефинициите

```
double x, y, z;
int m, n, p;
```

Кои от следните редици от символи са оператори за присвояване:

а) -x = y; б) x = -y; в) m + n = p;

г) p = x + y; д) z = x - y е) z = m + n;

ж) sin(0) = 0; з) x n + sin(z) к) 4 = sin(p + 5)?

В случаите а), в), ж) и к) редиците не са оператори за присвояване, тъй като на израз се присвоява израз. В случай г) редицата от символи е оператор за присвояване, но тъй като на цяла променлива се присвоява стойността на реален израз, компилаторът ще направи предупреждение за загуба на точност, а в случай з) е пропуснат символът '=' от знака за присвояване.

Задача 13. Да се напише програма, която въвежда стойности на реалните променливи a и b, след което разменя и извежда стойностите им (Например, ако a = 5.6, a b = -3.4, след изпълнението на програмата a да става -3.4 , a b да получава стойността 5.6).

Програма Zad13.cpp решава задачата.

Program Zad13.cpp

```
#include <iostream.h>
int main()
{cout << "a= ";
 double a;
 cin >> a;
 cout << "b= ";
 double b;
 cin >> b;
 double x;
 x = a;
 a = b;
 b = x;
 cout << "a= " << a << "\n";
 cout << "b =" << b << "\n";
 return 0;
}
```

В тази програма се използва работна променлива x, която съхранява първоначалната стойност на променливата a.

Задача 14. Да се напише програма, която въвежда положително трицифрено число и извежда на отделни редове цифрите на стотиците, на десетиците и на единиците на числото.

```
Програма Zad14.cpp решава задачата.
Program Zad14.cpp
#include <iostream.h>
#include <iomanip.h>
int main()
{ cout << "a - three-digit, integer and positive? ";
  int a ;
  cin >> a;
  short s, d, e;
  s = a / 100;
  d = a / 10 % 10;
  e = a % 10;
  cout << setw(10) << "stotici: " << setw(5) << s << "\n";
```

```
  cout << setw(10) << "desetici:" << setw(5) << d << "\n";
  cout << setw(10) << "edinici: " << setw(5) << e << "\n";
  return 0;
 }
```

Задача 15. На цялата променлива b да се присвои първата цифра на дробната част на положителното реално число x (Например, ако x = 52.467, то b = 4).

Програма Zad15.cpp решава задачата

```
Program Zad15.cpp
#include<iostream.h>
#include <math.h>
int main()
{ cout << "x>0? ";
  double x;
  cin >> x;
  int i = floor(x * 10);
  int b = i % 10;
  cout << x << "\n";
  cout << b << "\n";
  return 0;
}
```

Задача 16. Да се напише програма, която извежда 1, ако в записа на положителното четирицифрено число a, всички цифри са различни и 0 - в противен случай.

Програма Zad16.cpp решава задачата.

```
Program Zad20.cpp
#include <iostream.h>
int main()
{ cout << "a - four-digit, integer and positive? ";
  int a ;
  cin >> a;
  short h, s, d, e;
```

```
  h = a / 1000;
  s = a / 100 % 10;
  d = a / 10 % 10;
  e = a % 10;
  cout << (h != s && h != d && h != e &&
          s != d && s != e && d != e) << "\n";
  return 0;
}
```

4.2. Празен оператор

Това е най-простия оператор на C++. Описанието му е дадено на Фиг. 1.

Синтаксис

;

Операторът не съдържа никакви символи. Завършва със знака ;.

Семантика

Не извършва никакви действия. Използва се когато синтаксисът на някакъв оператор изисква присъствието на поне един оператор, а логиката на програмата не изисква такъв.

Фиг. 1.

Забележка: Излишни празни оператори не предизвикват грешка при компилация. Например, редицата от оператори

```
a = 150;;;
b = 50;;;;
c = a + b;;
```

се състои от: оператора за присвояване a = 150, 2 празни оператора, оператора за присвояване b = 50, 3 празни оператора, оператора за присвояване c = a + b и 1 празен оператор и е напълно допустим програмен фрагмент.

Други примери ще дадем по-късно.

4.3. Блок

Често синтаксисът на някакъв оператор на езика изисква използването на един оператор, а логиката на задачата – редица от оператори. В този случай се налага оформянето на блок (Фиг. 2.).

Синтаксис

{ <оператор$_1$>
  <оператор$_2$>
   . . .
  <оператор$_n$>
}

Семантика

Обединява няколко оператора в един, наречен блок. Може да бъде поставен навсякъде, където по синтаксис стои оператор.

Дефинициите в блока, се отнасят само за него, т.е. не могат да се използват извън него.

Фиг. 2.

Пример: Операторът

```
{cout << “a= “;
 double a;
 cin >> a;
 cout << “b= “;
 double b;
 cin >> b;
 double c = (a+b)/2;
 cout << “average{a, b} = “ << c << “\n”;
}
```

е блок. Опитът за използване на променливите a, b и c след блока, предизвиква грешка.

Препоръка: Двойката фигурни скобки, отварящи и затварящи блока да се поставят една под друга.

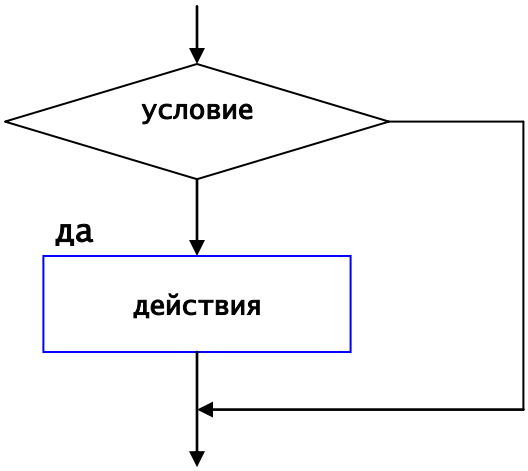
Забележка: За разлика от другите оператори, блокът не завършва със знака ;.

4.4. Условни оператори

Чрез тези оператори се реализират разклоняващи се изчислителни процеси. Оператор, който дава възможност да се изпълни (или не) един или друг оператор в зависимост от някакво условие, се нарича **условен**. Ще разгледаме следните условни оператори: if, if/else и switch.

4.4.1. Условен оператор if

Чрез този условен оператор се реализира разклоняващ се изчислителен процес от вид, илюстриран на Фиг. 3.



Фиг. 3.

Ако указаното условие е в сила, изпълняват се определени действия, а ако не – тези действия се прескачат. И в двата случая след това се изпълняват общи действия.

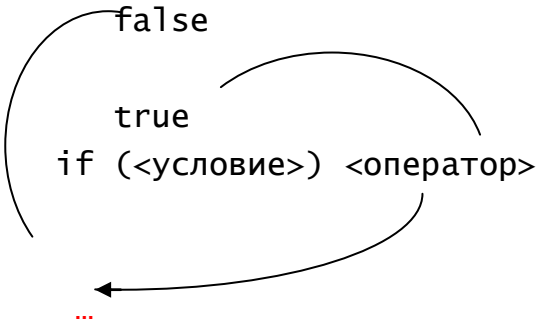
Условието се задава чрез някакъв булев израз, а действията – чрез оператор. Фиг. 4 описва подробно синтаксиса и семантиката на този оператор.

Синтаксис
if (<условие>) <оператор>
където

- if (ако) е запазена дума
- <условие> е булев израз;
- <оператор> е произволен оператор.

Семантика

Пресмята се стойността на булевия израз, представящ условието. Ако резултатът е true, изпълнява се <оператор>. В противен случай <оператор> не се изпълнява, т.е.



Фиг. 4.

Забележки:

1. Булевият израз, определящ <условие>, трябва да бъде определен. Огражда се в кръгли скобки.
2. Операторът след условието е точно един. Ако е необходимо няколко оператора да се изпълнят, трябва да се обединят в блок.

Задача 17. Да се напише програма, която намира май-малкото от три дадени реални числа.

Ще реализираме следните стъпки:

а) Въвеждане на стойности на реалните променливи a, b и c.

б) Инициализиране със стойността на a на работна реална променлива min, която ще играе и ролята на изходна променлива.

в) Сравняване на b с min. Ако стойността на b е по-малка от запомнения в min текущ минимум, запомня се b в min. В противен случай, min не се променя. Така min съдържа min{a, b}.

г) Сравняване на c с min. Ако стойността на c е по-малка от запомнения в min текущ минимум, c се запомня в min. В противен случай, min не се променя. Така min съдържа min{a, b, c}.

д) Извеждане на резултата – стойността на min.

Програма Zad17.cpp реализира този алгоритъм.

```
Program Zad17.cpp
#include <iostream.h>
int main()
{cout << "a= ";
 double a;
 cin >> a;
 cout << "b= ";
 double b;
 cin >> b;
 cout << "c= ";
 double c;
 cin >> c;
 double min = a;
 if (b < min) min = b;
 if (c < min) min = c;
 cout << "min{" << a << ", " << b << ", "
      << c << "}= " << min << "\n";
 return 0;
}
```

Ако вместо очаквано реално число, при въвеждане на стойности за променливите a, b и c, се въведе произволен низ, не представляващ число, буферът на клавиатурата, свързан със cin ще изпадне в състояние fail, а обектът cin ще има стойност false. Добрият стил за програмиране изисква в такъв случай програмата да прекъсне изпълнението си с подходящо съобщение за грешка. Програмата от задача 18 реализира този стил.

Задача 18. Да се напише програма, която намира най-малкото от три дадени реални числа. Програмата да извършва проверка за коректност на входните данни.

Програма Zad18.cpp решава задачата.

```
Program Zad18.cpp
#include <iostream.h>
int main()
{cout << "a= ";
 double a;
 cin >> a;
 if (!cin)
 {cout << "Error, bad input \n";
  return 1;
 }
 cout << "b= ";
 double b;
 cin >> b;
 if (!cin)
 {cout << "Error, bad input \n";
  return 1;
 }
 cout << "c= ";
 double c;
 cin >> c;
 if (!cin)
 {cout << "Error, bad input \n";
  return 1;
 }
 double min = a;
 if (b < min) min = b;
 if (c < min) min = c;
 cout << "min{" << a << ", " << b << ", "
      << c << "}= " << min << "\n";
 return 0;
}
```

Ще напомним, че операторът return предизвиква преустановяване работата на програмата, а стойността 1 – че е възникнала грешка.

Задача 19. да се сортира във възходящ ред редица от три реални числа, запомнени в променливите a, b и c.

Ще реализираме следните стъпки:

а) Въвеждане на стойности за a, b и c.

б) Сравняване на стойностите на a и b. Ако е в сила релацията b<a, извършва се размяна на стойностите на a и b. В противен случай – размяната не се извършва.

в) Сравняване на стойностите на a и c. Ако е в сила релацията c<a, извършва се размяна на стойностите на a и c. В противен случай – размяната не се извършва. След това действие, променливата a съдържа най-малката стойност на редицата.

г) Сравняване на стойностите на b и c. Ако е в сила релацията c<b, извършва се размяна на стойностите на b и c. В противен случай – размяната не се извършва.

е) Извеждане на стойностите на a, b, и c.

Програма Zad19.cpp реализира това описание.

```
Program Zad19.cpp
#include <iostream.h>
#include <iomanip.h>
int main()
{ cout << "a= ";
  double a;
  cin >> a;
  if (!cin)
  {cout << "Error, bad input \n";
   return 1;
  }
  cout << "b= ";
  double b;
  cin >> b;
  if (!cin)
  {cout << "Error, bad input \n";
   return 1;
  }
```

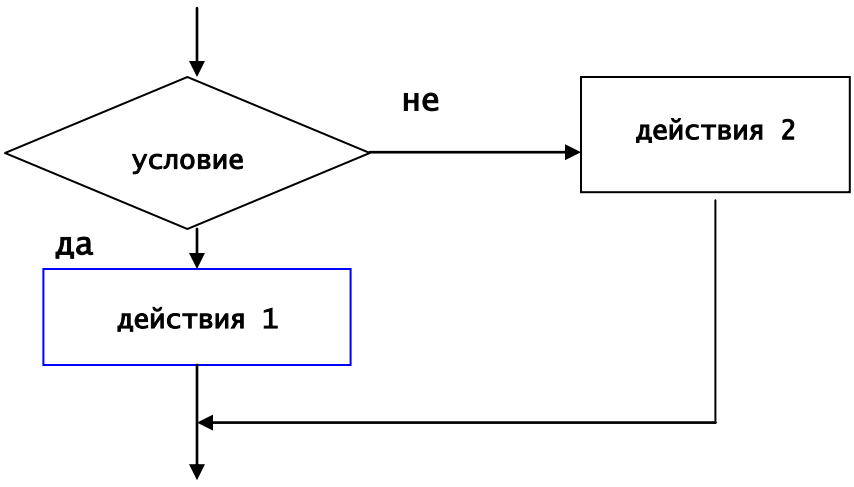
```
    cout << "c= ";
    double c;
    cin >> c;
    if (!cin)
    {cout << "Error, bad input \n";
     return 1;
    }
    if (b < a) {double x = a; a = b; b = x;}
    if (c < a) {double x = c; c = a; a = x;}
    if (c < b) {double x = c; c = b; b = x;}
    cout << setprecision(2) << setiosflags(ios :: fixed);
    cout << setw(10) << a << setw(10) << b << setw(10)
         << c << "\n";
    return 0;
}
```

Забележка: Променливата x е видима (може да се използва) само в блоковете, където е дефинирана.

4.4.2. Оператор if/else

Операторът се използва за избор на една от две възможни алтернативи в зависимост от стойността на дадено условие.

Чрез него се реализира разклоняващ се изчислителен процес от вид, илюстриран на Фиг. 5.



Фиг. 5.

Ако указаното условие е в сила, се изпълняват се едни действия, а ако не – други действия. И в двата случая след това се изпълняват общи действия.

Условието се задава чрез някакъв булев израз, а действия 1 и действия 2 – чрез оператори. Фиг. 6 описва подробно синтаксиса и семантиката на този оператор.

Синтаксис

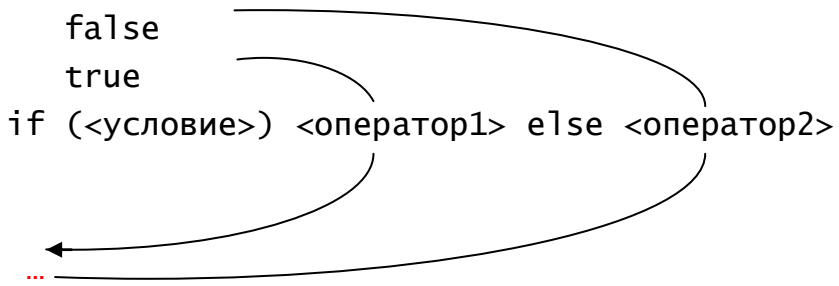
if (<условие>) <оператор1> else <оператор2>

където

- if (ако) и else (иначе) са запазени думи
- <оператор1> и <оператор2> са произволни оператори;
- <условие> е булев израз.

Семантика

Пресмята се стойността на булевия израз, представящ условието. Ако резултатът е true, изпълнява се <оператор1>. В противен случай се изпълнява <оператор2>, т.е.



Фиг. 6.

Забележки:

1. Булевият израз, определящ <условие>, трябва да бъде напълно определен. Задължително се огражда в кръгли скобки.

2. Операторът след условието е точно един. Ако е необходимо няколко оператора да се изпълнят, трябва да се обединят в блок.

3. Операторът след else е точно един. Ако е необходимо няколко оператора да се изпълнят, трябва да се обединят в блок.

Задача 20. Променливата y зависи от променливата x. Зависимостта е следната:

$$y = \begin{cases} \lg(x) + 1.82, & \text{ако } x \geq 1 \\ .^2 + 7.x + 8.82, & \text{ако } x < . \end{cases}$$

Да се напише програма, която по дадено x намира съответната стойност на y.

Програма Zad20.cpp решава задачата.

```
Program Zad20.cpp
#include <iostream.h>
#include <iomanip.h>
#include <math.h>
int main()
{ cout << "x= ";
  double x;
  cin >> x;
  if (!cin)
  {cout << "Error, bad input!!! \n";
   return 1;
  }
  double y;
  if (x >= 1) y = log10(x) + 1.82;
  else y = x*x - 7*x + 8.82;
  cout << setprecision(3) << setiosflags(ios :: fixed);
  cout << setw(10) << x << setw(10) << y << "\n";
  return 0;
}
```

След въвеждането на стойността на променливата x, програмата извършва проверка за валидност на въведената стойност. Изпълнението на оператора if/else води до пресмятане на стойността на булевия израз x >= 1. Ако тя е true, се изпълнява операторът за присвояване y = log10(x) + 1.82;. В противен случай се изпълнява операторът за присвояване y = x*x - 7*x + 8.82;, след което се извежда резултатът.

Вложени условни оператори

В условните оператори:

if (<условие>) <оператор>

if (<условие>) <оператор1> else <оператор2>

<оператор>, <оператор1> и <оператор2> са произволни оператори, в т. число могат да бъдат условни оператори. В този случай имаме вложени условни оператори.

При влагането е възможно да възникнат двусмислици. Ако в един условен оператор има повече запазени думи if отколкото else, възниква въпросът, за кой от операторите if се отнася съответното else. Например, нека разгледаме оператора

```
if (x >= 0) if ( x >= 5) x = 1/x; else x = -x;
```

Възможни са следните две различни тълкувания на този оператор:

а) if оператор, тялото на който е if/else оператор, т.е.

```
if (x >= 0)
   if (x >= 5) x = 1/x; else x = -x;
```

При това тълкувание, ако преди изпълнението на if оператора x има стойност –5, след изпълнението му, стойността на x остава непроменена.

б) if/else оператор, с if оператор след <условие>, т.е.

```
if (x >= 0) if ( x >= 5) x = 1/x;
else x = -x;
```

При това тълкувание, ако преди изпълнението на if/else оператора x има стойност –5, след изпълнението му, стойността на x става 5.

Записът чрез съответни подравнявания, не влияе на компилатора. В езика C++ има правило, което определя начина по който се изпълняват вложени условни оператори.

Правило: Всяко else се съчетава в един условен оператор с най-близкото преди него несъчетано if. Текстът се гледа отляво надясно.

Според това правило, компилаторът на C++ ще приеме първото тълкувание за горните вложени условни оператори.

Препоръка: Условен оператор да се влага в друг условен оператор само след else. Ако се налага да се вложи след условието, вложеният условен оператор да се направи блок.

Задачи върху операторите if и if/else

Задача 21. Ако променливата a има стойност 8, определете каква стойност ще има променливата b след изпълнението на оператора

```
if (a > 4) b = 5; else
  if (a < 4) b = -5; else
     if (a == 8) b = 8; else b = 3;
```

Тъй като е в сила условието a > 4, променливата b ще получи стойността 5.

Задача 22. Стойността на y зависи от x. Зависимостта е следната:

$$y = \begin{cases} x, & \text{ако } x \le 2 \\ 2, & \text{ако } x \in (2, 3] \\ -1, & \text{ако } x > 3 \end{cases}$$

Да се напише програма, която по дадено x, намира стойността на y.

Програма Zad22.cpp решава задачата.

```
Program Zad22.cpp
#include <iostream.h>
#include <iomanip.h>
int main()
{ cout << "x= ";
  double x;
  cin >> x;
  if (!cin)
  { cout << "Error, Bad input\n";
    return 1;
  }
  double y;
  if (x <= 2) y = x; else
     if (x <= 3) y = 2; else y = x-1;
```

```
  cout << setprecision(3) << setiosflags(ios :: fixed);
  cout << setw(10) << x << setw(10) << y << "\n";
  return 0;
 }
```

Забележка: В програма Zad22.cpp след първото else е в сила условието x > 2. Затова не е нужно то да се проверява.

Задача 23. Да се напише програма, която въвежда три реални числа a, b и c и извежда 0, ако не съществува триъгълник със страни a, b и c. Ако такъв триъгълник съществува, да извежда 3, 2 или 1 в зависимост от това какъв е триъгълникът - равностранен, равнобедрен или разностранен съответно.

Програма Zad23.cpp решава задачата.

```
Program Zad23.cpp
#include <iostream.h>
int main()
{ cout << "a= ";
  double a;
  cin >> a;
  if (!cin)
  {cout << "Error, bad input \n";
   return 1;
  }
  cout << "b= ";
  double b;
  cin >> b;
  if (!cin)
   {cout << "Error, bad input \n";
    return 1;
   }
  cout << "c= ";
  double c;
  cin >> c;
  if (!cin)
   {cout << "Error, bad input \n";
```

```
      return 1;
     }
    bool x = a <= 0 || b <= 0 || c <= 0 ||
             a+b <= c || a+c <= b || b+c <= a;
     if (x) cout << 0 << "\n"; else
         if (a == b && b == c) cout << 3 << "\n"; else
          if (a == b || a == c || b == c) cout << 2 << "\n"; else
             cout << 1 << "\n";
    return 0;
   }
```

Булевата променлива x е помощна. Тя има стойност true, ако a, b и c не са страни на триъгълник. Получена е след прилагане отрицание на условието a, b и c да са страни на триъгълник, т.е. на условието

a > 0 && b > 0 && c > 0 && a + b > c && a + c > b && b + c > a

като са използвани законите на де Морган.

Закони на де Морган:

!!A е еквивалентно на A

!(A || B) е еквивалентно на !A && !B

!(A && B) е еквивалентно на !A || !B

Задача 24. Да се напише програма, която на цялата променлива k присвоява номера на квадранта, в който се намира точка с координати (x, y). Точката не лежи на координатните оси, т.е. x.y ≠ 0.

Програмата Zad24.cpp решава задачата.

Program Zad24.cpp

```
#include <iostream.h>
int main()
{ cout << "x=";
  double x;
  cin >> x;
  if (!cin)
  {cout << "Error, Bad input! \n";
   return 1;
  }
```

```
cout << "y=";
double y;
cin >> y;
if (!cin)
{cout << "Error, Bad input! \n";
 return 1;
}
if (x*y == 0)
{cout << "The input is incorrect! \n";
 return 1;
}
int k;
if (x*y>0) {if (x>0) k = 1; else k= 3;}
else
   if (x>0) k = 4; else k = 2;
cout << "The point is in: " << k << "\n";
return 0;
}
```

4.4.3. Оператор switch

Често се налага да се избере за изпълнение един измежду множество от варианти. Пример за това дава следната задача.

Задача 25. Да се напише програма, която въвежда цифра, след което я извежда с думи.

За решаването на задачата трябва да се реализира следната неелементарна функция:

$$f = \begin{cases} \text{зведи zero}, & \text{ако въведената цифра е 0} \\ \text{зведи one}, & \text{ако въведената цифра е 1} \\ \quad \ldots & \quad \ldots \\ \text{зведи nine}, & \text{ако въведената цифра е 9} \end{cases}$$

Последното може да стане чрез следната програма:

```
#include <iostream.h>
int main()
{cout << "i= ";
 int i;
 cin >> i;
 if (!cin)
 {cout << "Error, bad input!\n";
  return 1;
 }
 if (i < 0 || i > 9)
 {cout << "Bad input \n";
  return 1;
 }
 else
 if (i == 0) cout << "zero \n";
 else if (i == 1) cout << "one \n";
 else if (i == 2) cout << "two \n";
 else if (i == 3) cout << "three \n";
 else if (i == 4) cout << "four \n";
 else if (i == 5) cout << "five \n";
 else if (i == 6) cout << "six \n";
 else if (i == 7) cout << "seven \n";
 else if (i == 8) cout << "eight \n";
 else if (i == 9) cout << "nine \n";
 return 0;
}
```

В нея са използвани вложени if и if/else оператори, условията на които сравняват променливата i с цифрите 0, 1, 2, …, 9.

Има по-удобна форма за реализиране на това влагане. Постига се чрез оператора за избор на вариант switch.

Програма Zad25.cpp е друго решение на задачата.

```
Program Zad25.cpp
#include <iostream.h>
int main()
{cout << "i= ";
 int i;
 cin >> i;
 if (!cin)
 {cout << "Error, bad input!\n";
  return 1;
 }
 if (i < 0 || i > 9)
 {cout << "Bad input \n";
  return 1;
 }
 else
 switch (i)
 {case 0 : cout << "zero \n"; break;
  case 1 : cout << "one \n"; break;
  case 2 : cout << "two \n"; break;
  case 3 : cout << "three \n"; break;
  case 4 : cout << "four \n"; break;
  case 5 : cout << "five \n"; break;
  case 6 : cout << "six \n"; break;
  case 7 : cout << "seven \n"; break;
  case 8 : cout << "eight \n"; break;
  case 9 : cout << "nine \n"; break;
 }
 return 0;
}
```

Операторът switch започва със запазената дума switch (ключ), следван от, ограден в кръгли скобки, цял израз. Между фигурните скобки са изброени вариантите на оператора. Описанието им започва със

запазената дума case (случай, вариант), следвана в случая от цифра, наречена етикет, двоеточие и редица от оператори.

Изпълнение на програмата

След въвеждането на стойност на променливата i се извършва проверка за коректност на въведеното. Нека въведената стойност е 7. Изпълнението на оператора switch причинява да бъде пресметната стойността на израза i – в случая 7. След това последователно сравнява тази стойност със стойностите на етикетите до намиране на етикета 7 и изпълнява редицата от оператори след него. В резултат върху екрана се извежда

seven

курсурът се премества на нов ред и се прекъсва изпълнението на оператора switch. Последното е причинено от оператора break в края на редицата от оператори за варианта с етикет 7.

Операторът switch реализира избор на вариант от множество варианти (възможности). Синтаксисът и семантиката му са дадени на Фиг. 7.

Синтаксис

switch (<израз>)
{ case <израз$_1$> : <редица_от_оператори$_1$>
  case <израз$_2$> : <редица_от_оператори$_2$>
   …
  case <израз$_{n-1}$> : <редица_от_оператори$_{n-1}$>
  [default : <редица_от_оператори$_n$>]
}

където

- switch (ключ), case (случай, избор или вариант) и default (по премълчаване) са запазени думи на езика;
- <израз> е израз от допустим тип (Типовете bool, int и char са допустими, реалните типове double и float не са допустими). Ще го наричаме още switch-израз.
- <израз$_1$>, <израз$_2$>, …, <израз$_{n-1}$> са константни изрази, задължително с различни стойности.

- <редица_от_оператори$_i$> (i = 1, 2, …, n) се дефинира по следния начин:

<редица_от_оператори> ::= <празно>|
<оператор>|
<оператор><редица_от_оператори>

Семантика

Намира се стойността на switch-израза. Получената константа се сравнява последователно със стойностите на етикетите <израз$_1$>, <израз$_2$>, … При съвпадение, се изпълняват операторите на съответния вариант и операторите на всички варианти, разположени след него, до срещане на оператор break. В противен случай, ако участва default-вариант, се изпълнява редицата от оператори, която му съответства и в случай, че не участва такъв – не следват никакви действия от оператора switch.

Фиг. 7.

Между фигурните скобки са изброени вариантите на оператора. Всеки вариант (без евентуално един) започва със запазената дума case, следвана от израз (нарича се още case-израз или етикет), който се пресмята по време на компилация. Такива изрази се наричат **константни**. Те не зависят от входните данни. След константния израз се поставя знакът двоеточие, следван от редица от оператори (оператори на варианта), която може да е празна. Сред вариантите може да има един (не е задължителен), който няма case-израз и започва със запазената дума default. Той се изпълнява в случай, че никой от останалите варианти не е бил изпълнен.

Съществува възможност програмистът да съобщи на компилатора, че желае да се изпълни само редицата от оператори на варианта с етикет, съвпадащ със стойността на switch-израза, а не и всички следващи го. Това се реализира чрез използване на оператор break в края на редицата от оператори на варианта. Този оператор предизвиква прекъсване на изпълнението на оператора switch и предаване на управлението на първия оператор след него (Фиг. 8.).

Операторът break принадлежи към групата на т. нар. **оператори за преход**. Тези оператори предават управлението безусловно в някаква точка на програмата.

Синтаксис

break;

Семантика

Прекратява изпълнеието на най-вътрешния съдържащ го оператор switch или оператор за цикъл. Изпълнението на програмата продължава от оператора, следващ (съдържащ) прекъснатия.

Фиг. 8.

Програмистът съзнателно пропуска оператора break, когато за няколко различни стойности от множеството от стойности, трябва да се извършат еднакви действия.

Забележка: Ако в оператора switch не е използван операторът break, ще бъде изпълнена редицата от оператори на варианта, чийто case-израз съвпада със стойността на switch-израза и също всички след него.

Използването на оператора switch има едно единствено предимство пред операторите if и if/else – прави реализацията по-ясна. Основен негов недостатък е, че може да се прилага при много специални обстоятелства, произтичащи от наложените ограничения на типа на switch-израза, а именно, той трябва да е цял, булев или символен. Освен това, използването на оператора break, затруднява доказването на важни математически свойства на програмите, използващи break.

Задачи върху оператора switch

Задача 26. Да се напише програма, която по зададено реално число x намира стойността на един от следните изрази:

y = x - 5

y = sin(x)

y = cos(x)
y = exp(x).

Изборът на желания израз да става по следния начин: при въвеждане на цифрата 1 се избира първият, на 2 – вторият, на 3 – третият и на 4 – четвъртия израз.

Програма Zad25.cpp решава задачата.

```
#include <iostream.h>
#include <iomanip.h>
#include <math.h>
int main()
{cout << "================================\n";
 cout << "|    y = x-5             -> 1    |\n";
 cout << "|    y = sin(x)          -> 2    |\n ";
 cout << "|    y = cos(x)          -> 3    |\n";
 cout << "|    y = exp(x)          -> 4    |\n";
 cout << "================================\n";
 cout << " 1, 2, 3 or 4? \n";
 int i;
 cin >> i;
 if (!cin)
 {cout << "Error, Bad input!!! \n";
  return 1;
 }
 if (i == 1 || i == 2 || i == 3 || i == 4)
 {cout << "x= ";
  double x;
  cin >> x;
  if (!cin)
  {cout << "Error, Bad input!! \n";
   return 1;
  }
  double y;
  switch (i)
  {case 1: y = x - 5;  break;
   case 2: y = sin(x); break;
```

```
    case 3: y = cos(x); break;
    case 4: y = exp(x); break;
   }
   cout << "y= " << y << "\n";
  }
  else
  {cout << "Error, Bad choise!!\n";
   return 1;
  }
  return 0;
 }
```

Задачи

Задача 1. Явява ли се условен оператор редицата от символи:

а) if (x < y) x = 0; else y = 0;

б) if (x > y) x = 0; else cin >> y;

в) if (x >= y) x = 0; y = 0; else cout << z;

г) if (x < y) ; else z = 5;

д) if (x < y < z) then z = z + 1;

е) if (x != y) z = z+1; x = x + y;

Задача 2. Кое условие е в сила след запазената дума else на условния оператор:

а) if (a > 1 && a < 5) b = 5; else b = 10;

б) if (a < 1 || a > 5) b = a; else a = b;

в) if (a = b || a = c || b = c) c = a + b; else c = a – b;

Задача 3. Да се намерят грешките в следните оператори:

а) if (1 < x < 2) x = x + 1; y = 0;
   else x = 0; y = y + 1;

б) if (1 < x) && (x < 2)
   {x = x + 1;
    y = 0;
    };
   else
   {x = 0;

```
    y = y + 1;
  };
```

Задача 4. Да се напише програма, която по дадено реално число x намира стойността на y, където

$$y = \begin{cases} , & \text{ако } x \le 0 \\ ^2, & \text{ако } x > 0 \end{cases}$$

Задача 5. Да се напише програма, която по зададени стойности на реалните променливи a, b и c намира:

а) min{a+b+c, a.b.c} + 15.2

б) max{a^2 - b^3 + c, a – 17.3 b, 3.1 a + 3.5 b – 8 c} - 17.9.

Задача 6. Да се напише програма, която увеличава по-малкото от две дадени цели неравни числа пет пъти, а по-голямото число намалява 8 пъти.

Задача 7. Да се напише програма, която въвежда четири реални числа и ги извежда във възходящ (низходящ) ред върху екрана.

Задача 8. Дадени са три числа a, b и c. Да се напише програма, в резултат от изпълнението на която, ако е в сила релацията a ≥ b ≥ c, числата се удвояват, в противен случай числата се заменят с техните абсолютни стойности.

Задача 9. Да се намери стойността на z след изпълнението на операторите

```
z = 0;
if (x > 0) if (y > 0) z = 1; else z = 2;
```

ако:

а) x = y = 1 б) x = 1, y = -1 в) x = -1, y = 1

Задача 10. Да се запише указаното действие чрез един условен оператор:

а) $d = \begin{cases} \text{ax}\{a, b\} & \text{ако } x < 0 \\ \text{in}(a, b) & \text{ако } x \ge 0 \end{cases}$

б) d = max(a, b, c)

Задача 11. Да се напише условен оператор, който е еквивалентен на оператора за присвояване

```
x =  a || b && c;
```

където всички променливи са булеви и в който не се използват логически операции (Например, операторът x = not a; е еквивалентен на оператора if (a) x = false; else x = true;).

Задача 12. Да се напише оператор за присвояване, еквивалентен на условния оператор

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

(всички променливи са булеви).

Задача 13. Да се напише програма, която по зададено число a, намира корена на уравнението f(x) = 0, където

$$f(x) = \begin{cases} xa^{\frac{1}{3}} + |a-1|^{\frac{1}{2}}, & \text{ако } a > \\ e^{ax} - a^2 - 5, & \text{ако } a \leq \end{cases}$$