Тема 5
Аритметични оператори и логически оператори. Пресмятане на изрази

1. Аритметични оператори

В програмните езици като елементи от програмите са предвидени оператори, чрез които се извършват различни математически, логически или други действия (операции) върху данните. Език C/C++ поддържа аритметични и логически операции, сравнения (отношения) и побитови операции. Данните, необходими за извършване на дадена операция (оператор) се наричат операнди. В зависимост от броя на операндите операторите се разделят на ***унарни*** и ***бинарни***. Бинарните оператори използват два операнда, докато в унарните участва само един операнд.

Бинарни аритметични оператори в език C/C++ са:

- събиране (+);
- изваждане (-);
- умножение (*);
- деление (/);
- деление по модул (%).

Като оператори ***събиране***, ***изваждане*** и ***умножение*** са напълно идентични със съответните математическите операции. Особеност на оператор ***деление*** е, че когато операндите са реални числа, резултатът също е реално число, докато при деление на целочислени числа, резултатът е цяло число т.е. делението е целочислено. Пример:

```
int x=2, y=9, z;
float p=2, q=9, r;
z=y/x;       // резултат z=4
r=q/p;       // резултат r=4.5
```

Резултатът от оператор ***% (деление по модул)*** е остатък от целочислено деление. Пример:

```
int x=2, y=6, z;
int p=3, q=5, r;
z=y%x;       // резултат z=0
r=q%p;       // резултат r=2
```

Оператор ***деление по модул*** се използва само за целочислени стойности и е неприложим за числа от тип ***float*** и ***double***.

Унарни аритметични оператори в C/C++ са:

- унарен + (запазване на знака на числото);

- унарен - (промяна на знака на числото);
- инкрементиране ++ (увеличаване на стойността с 1);
- декрементиране -- (намаляване на стойността с 1).

Съществуват две форми на операторите инкрементиране и декрементиране:

- префиксна ++i, --i;
- суфиксна i++, i--.

Разликата между двете форми е, че при префиксната форма първо се инкрементира (декрементира) стойността, след което се изпълнява другият оператор. При суфиксната форма е обратно – инкрементирането (декрементирането) се изпълнява след другият оператор. В следващият пример оператор инкрементиране (++) се използва съвместно с оператор присвояване (=).

```
int x=15;
int y;
y=++x;
// резултантните стойности са: y=16; x=16.

int x=15;
int y;
y=x++;
// резултантните стойности са: y=15; x=16.
```

В резултат, при първият случай, когато се използва префиксната форма на инкрементирането, стойностите на променливите x и y са равни, тъй като стойността на първо се увеличава с единица и след това се присвоява на променливата y. Във втория, стойността на променливата x първо се присвоята на y и едва след това се инкрементира.

2. Логически оператори и отношения

В булевата алгебра резултатът от изпълнението на сравнения (по-голямо, по-малко, равно, различно и др.) и логически операции (логическо И, логическо ИЛИ, логическо НЕ) е логически израз, чиято стойност може да е вярно твърдение (***true***) или невярно твърдение (***false***). В програмните езици са предвидени оператори, които да изпълняват тези операции. Резултатът от тяхното изпълнение е от логически тип. За разлика от други езици за програмиране, първоначално в език C не е предвиден логически тип данни. За вярно и невярно твърдение се използват аритметичните стойности: 0 или различно от 0. Ако отношението не е вярно, то стойността на резултата е 0, което съответства на логическа стойност ***true***. Ако отношението е вярно, стойността на резултата е различна от 0, което съответства на логическа стойност ***false***. В последствие в езика е добавен булевия тип ***bool***. MS Visual C++ 5.0 и следващите версии поддържат вградения тип ***bool***, чиито константни стойности са две: ***true*** и ***false***.

В език C/C++ са предвидени следните оператори за сравнения:

> по-голямо

>= по-голямо или равно

< по-малко

<= по-малко или равно

== равно

!= различно

Логическите оператори в C и C++ са:

&& логическо И

|| логическо ИЛИ

! логическо НЕ

Действието на всеки логически оператор се представя с т.нар. ***таблица на истинност***. В нея се представят резултатите от изпълнението на оператора при всяка една различна комбинация от входни данни.

Таблицата на истинност на логически оператор И е следната:

x	y	x&&y
false	false	false
false	true	false
true	false	false
true	true	true

Таблицата на истинност на логически оператор ИЛИ е следната:

x	y	x||y
false	false	false
false	true	true
true	false	true
true	true	true

Таблицата на истинност на логически оператор НЕ е следната:

x	!x
false	true
true	false

В случая, в таблиците се използват стойности ***true*** и ***false*** за *вярно* и *невярно твърдение*. Както беше споменато, в C/C++ стойностите **!0** (различно от нула) и **0** (нула) се интерпретират като true и false, съответно.

Примери за действието на логически оператори и отношения са следните: нека са дефинирани променливите a и b като са инициализирани със стойности 4 и 7:

int a=4,b=7;

При изпълнението на изразите се получава следното:

Резултатът от изпълнение на израз **a<0** е false

Резултатът от изпълнение на израз **b>=0** е true

Резултатът от изпълнение на израз **(a>b)&&(a>0)** е false

Резултатът от изпълнение на израз **!a** е false

Резултатът от изпълнение на израз **(a<0)||((b>0)** е true

Логическите изрази и отношения се използват при задаване на условия в конструкциите, които реализират цикли и разклонения и се записват в програмния код, където е необходимо.

Тема 6
Реализация на разклонени алгоритми. Управляващи конструкции в език C/C++

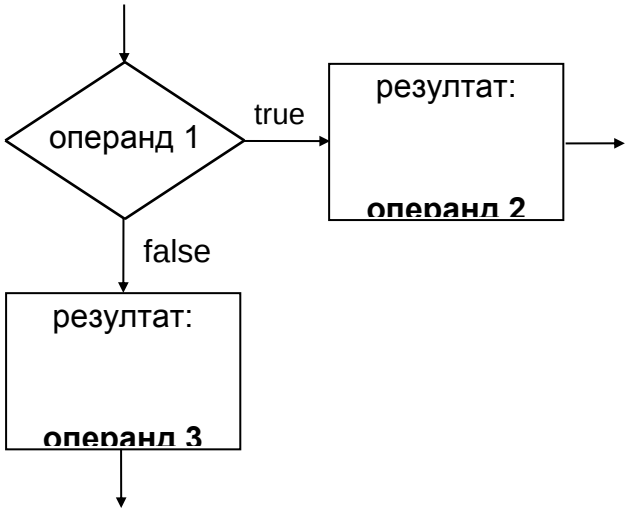
1. Оператор за условен израз (*?:*)

C/C++ разполага с редица полезни оператори, които повишават ефективността на програмирането. Такав е ***операторът за условен израз***. Чрез този оператор могат да се реализират елементарни разклонения в програмите без да е необходимо да се използват управляващите конструкции.

Операторът за условен израз има следния синтаксис:

операнд 1 ? операнд 2: операнд 3;

Операндите могат да са: *константи, променливи или изрази*. Опрераторът изисква три операнда и може да се срещне и с името „троен оператор". Неговото действие е следното: пресмята се *операнд 1*; ако резултатът е различен от 0 или е стойност true (т.е. резултатът е верен), изчислява се резултатът от *операнд 2* и неговата стойност е полученият резултат. Ако резултатът от изпълнението на *операнд 1* е равен на 0 или false (т.е. резултатът не е верен), изчислява се *операнд 3* и стойността му е общият резултат от действието на оператора. В блоков вид, действието на оператора за условен израз е представено на фиг. 7.1.



Фиг. 7.1. Конструкция на операцията за условен израз

Примери:

```
int x,y,z;
...
z=(x>y)?1:0;       // z приема стойност 1, ако изразът x>y е верен
z=(x>y)?x:y;       // z приема стойността на по-голямото число от x и y
```

```
    z=(x+y)? 10:(x/2);   /*ако x+y е различно от 0,то z=10, ако x+y е равно
на 0, то z=x/2 */
```

2. Управляващи конструкции

Всички езици за програмиране притежават множество конструкции, които задават реда на изпълнение на операторите, оформящи алгоритъма на програмата. Тези конструкции се наричат ***управляващи*.** Доброто познаване на управляващите структури е предпоставка за написване на добре структурирани програми. Към управляващите конструкции спадат тези за реализация на разклонени и циклични алгоритми.

2.1 Съставен оператор (оператор-блок) {…}

Съставният оператор е част от управляващите конструкции, въпреки че сам по себе си не реализира управление. Той представлява фрагмент от програма, разположен между фигурни скоби: **{...}**.

Съставният оператор е частен случай на ***блок***. Блоковете участват като елементи в много синтактични конструкции като интерпретацията им може да е различна. В общият случай, блокът представлява фрагмент от програма, който се интерпретира като едно цяло. В него се съдържат дефиниции и изпълними конструкции.

При управляващите конструкции, като ***if-else***, ***while-do,*** интерпретацията на оператора ***блок*** е като ***съставен оператор***. **Съставният оператор обединява множество конструкции в един** т.е. групата оператори се интерпретират като един оператор.

2.2. Условна конструкция *if*

Чрез конструкция ***if*** се прави избор от два възможни клона на програмата. Синтаксисът на конструкцията е следният:

if (условие) оператор;

Условието е израз, който се изчислява. Резултатът от изчислението е или ***true***, (всяка стойност, различна от 0), или ***false*** (стойност равна на 0). В случая, стойността ***true*** се интерпретира като вярно твърдение, а стойност ***false*** – като невярно твърдение. В общия случай условието може да е променлива или аритметичен израз, не само сравнение или логически израз, въпреки че при писането на програми последните са по-често използваните.

Операторът след скобите задължително е един. Ако алгоритъмът налага след условието да има повече от един оператор, те се обединяват с помощта на съставен оператор (блок). В този случай, ***if*** има следната конструкция:

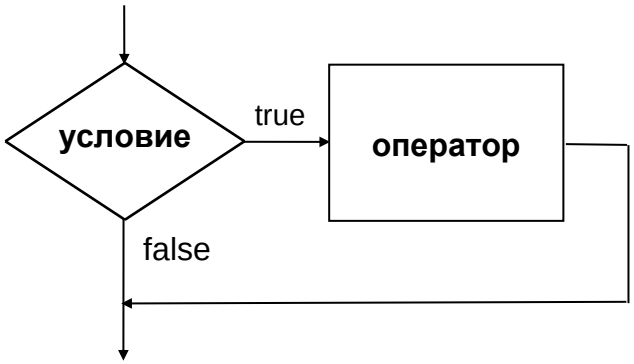
if (условие)

{

тяло на if-конструкцията

}

Действието на конструкцията е следното: пресмята се условието; ако резултатът е ***true***, изпълнява се тялото на if-конструкцията. При невярно твърдение, се продължава със следващата конструкция след ***if***. Действието на условната конструкцията ***if*** може да се представи с блоковата диаграма, показана на фиг. 7.2.



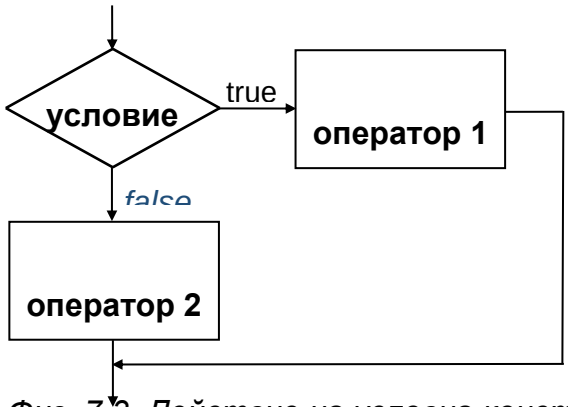
*Фиг. 7.2. Действие на условна конструкция **if***

2.3. Условна конструкция *if-else*

Конструкция ***if-else*** служи за избор на една от две възможни алтернативи, в зависимост от стойността на зададен условен израз. Общият вид на конструкцията е:

if (условие) оператор 1;

else оператор 2;



*Фиг. 7.3. Действие на условна конструкция **if-else***

Действието на конструкцията е следното: пресмята се условието; ако стойността на израза е ***true*** (или стойност различна от 0), изпълнява се *оператор 1*. Ако стойността на израза е ***false*** (или стойност равна на 0), изпълнява се *оператор 2*. Блоковата диаграма на фиг. 7.3 описва действието на конструкция ***if-else.***

Оператори 1 и 2 могат да са съставни. В този случай, ***if-else*** има следната конструкция:

if (условие)

{

Тяло на if-конструкцията

}

else

{

Тяло на else-конструкцията

}

Когато операторите след ***if*** и ***else*** са съставни, не се изисква след тях да се поставя символ **;** (раздел за край). Конструкциите ***if*** и ***if-else*** сами по себе си също се третират като една конструкция.

Действието на конструкции if и if-else може да бъде илюстрирано чрез следните примери:

Пример 1: Въвеждане на две числа и намиране на числото с по-голямата стойност.

```
#include <stdio.h>

main()
{
      int x, y, z;

      printf("Enter first number:");
      scanf("%d",&x);
      printf("Enter second number:");
      scanf("%d",&y);
      if(x>y) z=x;
      else z=y;
      printf("\nThe first number is %d",x);
      printf("\nThe second number is %d",y);
      printf("\nThe great number is %d",z);
      return 0;

}
```

Пример 2: Въвеждане на число и определяне дали числото е четно или нечетно. В условният израз проверката е чрез остатък от деление на стойност 2.

```
#include <stdio.h>
main()
{

      int x;

      printf("Enter a number:");
```

```
      scanf("%d",&x);
      if (x%2) printf("\nOdd number");
      if (!(x%2)) printf("\nEven number");
      return 0;
}
```

Пример 3: Въвеждане на три числа и намиране на по-голямата от трите стойности.

```
#include <stdio.h>

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

      printf("\n First value is: ");
      scanf("%d",&a);
      printf("\n Second value is: ");
      scanf("%d",&b);
      printf("\n Third value is: ");
      scanf("%d",&c);

      if( a > b) greater = a;
      else greater = b;
      if( c > greater) greater = c;

      printf("\n The great value is %d",greater);
      return 0;
}
```

2.4. Вложени if и if-else конструкции

Конструкциите if и if-else могат да бъдат ***влагани***, т.е. като оператори след условието и след ***else*** могат да бъдат поставени други конструкции ***if*** и ***if-else***. Пример за използване на тези управляващи конструкции и влагането им една в друга е програмата за пресмятане корените на квадратното уравнение.

```
// програма за пресмятане корените на квадратното уравнение

#include <stdio.h>

#include <math.h>

main()

{

      int a,b,c;

      double d,x1,x2;


      printf("\n a=");

      scanf("%d",&a);

      printf(" b=");
```

```
      scanf("%d",&b);
      printf(" c=");
      scanf("%d",&c);
      if(a!=0)           // проверка дали уравнението е квадратно
      {
            d=b*b+4*a*c;  // проверка дали корените са реални
            if(d<0) printf("Няма реални корени.");
            else
            {
                  if(d==0)    // проверка за един двоен корен
                  {
                    x1 = -b/(2*a);
                    printf("\n Един двоен корен x=%f",x1);
                  }
                  else
                  {
                    x1=(-b+sqrt(d))/(2*a);
                    x2=(-b-sqrt(d))/(2*a);
                    printf ("\n Корение на уравнението са:%f, %f", x1,
x2);
                        }
            }
      }
      else printf("Уравнението е линейно.");
      return 0;
}
```

2.5. Конструкция за избор *switch-case*

Чрез конструкция ***switch*** се осъществява избор от няколко възможни варианта. Общият вид на конструкцията е следния:

switch (израз)

{ case константенИзраз1: оператор 11;оператор 12;... break;

case константенИзраз2: оператор 21;оператор 22;...; break;

...

case константенИзраз n: оператр n1;оператор n2;...; break;

default : оператор 1; оператор 2;...;

}

Действието на конструкция ***switch*** е следното: пресмята се стойността на израза в скобите и се сравнява с константните изрази; ако има съвпадение на стойностите, изпълняват се операторите след съответния константен израз. Например, нека стойността на израза да е равна на константен израз 1. В този случай, се изпълняват *оператори 11, 12*, и т.н., докато не бъде срещнат оператор ***break*** или не бъде достигнат края на конструкцията ***switch***. Затова, за да се избегнат неточности в алгоритъма, последният от групата оператори след константите, трябва да е ***break***.

При конструкция ***switch*** е предвидена възможност, при несъвпадение с нито една от константите, да се изпълнят група оператори след ключова дума ***default***. Тази част в ***switch*** не е задължителна. Ако липсва ***default***-част в конструкцията ***switch*** и стойността на израза не съвпада с нито една от константите, не се изпълнява нито един от операторите в ***switch***. Действието на този оператор може да се представи чрез блоковата диаграма на фиг. 7.4.

*Фиг. 7.4. Действие на условна конструкция **switch-case***

В повечето случаи, ***switch*** може да се замени с няколко вложени конструкции ***if-else-if***. В общия случай, обаче конструкция ***switch*** не е еквивалентна на няколко вложени ***if-else-if*** конструкции поради следните особености:

- Условието за избор е само при съвпадение на стойностите на изразите при ***switch*** и ***case***, докато при структурата ***if-else-if*** условията могат да бъдат произволни.
- При ***switch*** се проверява съвпадението с константа. След ***case*** изразите могат да са само константни т.е. трябва да са изчислени по време на компилацията.
- Конструкция ***switch*** допуска наличие на празни варианти в част ***default***.

Конструкция ***switch*** е удобна, когато трябва да се направи избор от няколко възможни варианта. Примери за използване на ***switch*** е следващите програми.

```
#include <stdio.h>

main()
{
      int day;
      printf("\nEnter day (1..7):");
```

```
      scanf("%d",&day);

      switch(day)
      {
            case 1: printf("Monday");
                        break;
            case 2: printf("Tuesday");
                        break;
            case 3: printf("Wednesday");
                        break;
            case 4: printf("Thursday");
                        break;
            case 5: printf("Friday");
                        break;
            case 6: printf("Saturday");
                        break;
            case 7: printf("Sunday");
                        break;
            default: printf("The week has only seven days.");

      }
   return 0;
}
```

В примера според стойността на променливата **day** се извежда на екрана поредният ден от седмицата. Ако стойността на променливата не съвпада с нито една от константите от 1 до 7, тогава се изпълнява клаузата **default**.

2.6. Конструкция *break*

Конструкция ***break*** предизвиква завършване на конструкциите, които организират програмен цикъл (***while, do-while, for***) или конструкция ***switch***.

Общият вид на конструкцията е:

break;

За правилната реализация на алгоритми чрез конструкция ***switch*** се използва ***break***. Чрез него се гарантира, че след изпълнение на един от вариантите, останалите няма да бъдат проверявани т.е. ще бъде избран само един от възможните варианти. Конструкция ***break*** се използва и в итерационните конструкции (***while, do-while, for***). В този случай, тя предизвиква изход от цикъла, независимо от това дали е изпълнено или неизпълнено условието за изход.