Лабораторно упражнение № 2
Аритметични оператори. Реализация на задачи с линеен алгоритъм. Стандартни математически функции. Преобразуване на типове

I. Теоретична обосновка

Линейни алгоритми са тези, които запазват нормалната линейна последователност на изпълнение на операциите т.е. операциите се изпълняват по реда на тяхното записване. В програмните системи линейните алгоритми не се срещат в чист вид, а само като част от друг алгоритъм.

Линейните алгоритми обикновено се използват за математически изчисления. За програмна реализация на линейни алгоритми са необходими: пресмятане на аритметични изрази; оператор за присвояване; функции за въвеждане и извеждане на данни.

1. Аритметични оператори

Език С поддържа унарни и бинарни оператори. Бинарните използват от два операнда, докато в унарните участва само един операнд. Бинарни оператори са:

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

Унарни оператори 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).***

2. Изрази. Оператор за присвояване

Израз е правило за изчисляване на стойност т.е. всяка валидна за езика комбинация операции и операнди (данни), по която се пресмята стойност. В резултат на изчислението на даден израз се получава стойност.

Синтаксисът на оператора за присвояване е:

променлива = израз;

Действието на оператора е**: пресмята се стойността на израза отдясно и се присвоява на променливата отляво**.

В език С съществува ***съкратена форма на оператора за присвояване***:

операнд1 ***операция*****= операнд2;**

Тази форма е еквивалента на:

операнд1 = (операнд1) ***операция*** **(операнд2);**

Например, вместо *p=p+2;* записът е *p+=2;*

3. Стандартни математически функции

Тъй като аритметичните операции в програмните езици се ограничават до основните математически операции, за решаване на по-сложни математически задачи и

извършване на изчисления в програмите, към компилаторите се добавят библиотеки със стандартни математически функции. Такива са функциите за: пресмятане на квадратен корен, повдигане на степен, пресмятане на логаритъм, тригонометрични изчисления и др.

При компилаторите на C/C++, стандартните математически функции са дефинирани в заглавен файл ***math.h***. Затова, когато се използват тези функции в програмите, се включва заглавния файл:

#include <math.h>

Най-често използвани стандартни, математически функции са:

- ***sqrt()*** – пресмята квадратен корен от посочения в скобите аргумент

Функцията има следната декларация:

double sqrt(double x);

Фукнцията sqrt() изисква като аргумент константа, променлива или израз от тип double и връща резултат от същия тип. Ако аргумента не е от посочения тип, компилаторът се опитва да преобразува типа до double.

- ***pow()*** – използва се за повдигане на степен;

Функцията има следната декларация:

double pow(double x, double y);

Функцията връща като резултат, стойността на израза: $\mathbf{x^y}$. Аргументите са от тип double и функцията връща резултат от същия тип.

- ***exp()*** – пресмята стойността на израза $\mathbf{e^x}$;

Функцията има следната декларация:

double exp(double x);

Резултатът от функцията е пресмятане на израза $\mathbf{e^x}$, като **x** е аргумент на функцията. Аргументът и резултатът от функцията са от тип ***double*.**

- ***log()*** - пресмята стойността на израза **ln x**;

Функцията има следната декларация:

double log(double x);

Резултатът от функцията е пресмятане на израза **ln x**, като **x** е аргумент на функцията. Аргументът и резултатът от функцията са от тип ***double*.**

- ***log10()*** - пресмята стойността на израза **lg x**;

Функцията има следната декларация:

double log10(double x);

Резултатът от функцията е пресмятане на израза **lg x**, като **x** е аргумент на функцията. Аргументът и резултатът от функцията са от тип ***double*.**

- ***abs(), fabs()*** – функциите пресмятат модул от зададения аргумент;

Функциите има следните декларации:

int abs(int x);

double fabs(double x);

Действието на двете функции е идентично – пресмятат абсолютната стойност от зададения аргумент **x**. Разликата между двете функции е, че ***abs()*** изисква аргумент от тип ***int*** и връща резултат от същия тип, докато функция ***fabs()*** изисква аргумент от тип double и връща резултат от тип ***double*.**

- ***sin()*** – пресмята синус от зададения ъгъл.

Функцията има следната декларация:

double sin(double x);

Функцията пресмята **Sin(x)**, като аргументът **x** е ъгъл, зададен в радиани. Аргументът и резултатът от функцията са от тип ***double***.

- ***cos()*** – пресмята косинус от зададения ъгъл.

Функцията има следната декларация:

double cos(double x);

Функцията пресмята **Cos(x)**, като аргументът **x** е ъгъл, зададен в радиани. Аргументът и резултатът от функцията са от тип ***double***.

- ***tan()*** – пресмята тангенс от зададения ъгъл.

Функцията има следната декларация:

double tan(double x);

Функцията пресмята **tan(x)**, като аргументът **x** е ъгъл, зададен в радиани. Аргументът и резултатът от функцията са от тип ***double***.

4. Преобразуване на типове

Преобразуването на типове е често срещана операция в програмните езици. Преобразуването на типове е или ***явно***, чрез използване на специален оператор, или ***неявно*** – автоматично, по съответните правила. Явното преобразуване на типове е само в език C++.

4.1. Неявно преобразуване на типове

Неявното преобразуване на типове се осъществява автоматично в следните случаи:

- при изчисляване на изрази;
- при присвояване на стойност;
- при връщане стойности от функции.

Автоматичното преобразуване на типове не винаги е възможно. В случай, има несъответствие на типовете и не е възможно тяхното преобразуване, компилаторът издава съобщение за грешка.

4.2. Явно преобразуване на типове *(typecasting)*

Явното преобразуване на типове е чрез оператор ***привеждане (typecast)***.

Синтаксисът на оператора е:

(тип)израз;

Действието на оператор ***typecast*** е: привежда типа на израза към типа, указан в скобите.

II. Задачи за изпълнение

1. Да се състави програма за размяна на две числа.

Упътване: Задачата може да бъде решена по два начина:

- чрез използване на междинна променлива;

- без използване на междинна променлива.

2. Да се състави програма за преобразуване на температура от скалата на Фаренхайд в скала на Целзий.

Упътване: Формулата за преобразуване е: $C = \frac{F - 32}{1.8}$,

където C и F са температура по Целзий и Фаренхайт, съответно.

3. Да се състави програма за намиране на размерите на следните типове: int, char, float, double, short, long, signed, unsigned, unsigned char, long double.
Упътване: Да се използва оператор sizeof.

4. Да се състави програма за намиране на лице на триъгълник по зададени три страни. Стойностите на страните се задават от клавиатурата.

5. Да се състави програма, чрез която се извършва ротация на точка с координати x, y на определен ъгъл на завъртане. Стойностите на координати на точка и ъгъла на завъртане се въвеждат от клавиатурата.

Упътване: Ротацията на точка е графична трансформация, при която точката се завърта под определен ъгъл спрямо началото на координатната система. Новите координати на точката (x1, y1) се пресмятат по формулата:

$x1 = x\cos\alpha - y\sin\alpha$

$y1 = x\sin\alpha + y\cos\alpha$

като α е ъгълът на завъртане.

6. Да се състави програма за пресмятане на функцията $y=|x^5+\ln x-\lg x|$

Лабораторно упражнение № 3
Управляващи конструкции. Реализация на разклонени алгоритми

I. Теоретична обосновка

При ***разклонените алгоритми*** се нарушава нормалният, линейно-последователен ред на изпълнение на операциите в програмите т.е. операторите не се изпълняват по реда на тяхното записване в програмата. Реализацията на разклонени алгоритми е с помощта на т.нар. ***управляващи конструкции***. Чрез тях се задава реда на изпълнение на операторите и конструкциите, които формират алгоритъмът на програмата. Доброто познаване на управляващите структури е предпоставка за написване на добре структурирани програми.

1. Конструкция if

Чрез конструкция if се прави избор от два възможни клона на програмата.
Синтаксисът 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. Стойността TRUE се интерпретира като вярно твърдение, а стойност FALSE – като невярно твърдение.

Условието може да е променлива или аритметичен израз, не само сравнение или логически израз.

Действието на конструкцията е следното: Пресмята се условието. Ако резултатът е TRUE, изпълнява се операторът след условието. При невярно твърдение, се продължава със следващият оператор след ключова дума ***if***.

2. Конструкция if-else

Служи за избор на една от две възможни алтернативи, в зависимост от стойността на зададен условен израз.

Синтаксис:
if (условие) оператор 1;

else оператор 2;

Действието на конструкцията е следното: Пресмята се условието. Ако стойността на израза е различна от 0 (TRUE), изпълнява се оператор 1. Ако стойността на израза е равна на 0 (FALSE), изпълнява се оператор 2.

3. Конструкция за избор switch

Чрез оператор switch се осъществява избор от няколко възможни варианта.
Синтаксис:

```
switch (израз)
{
  case   константен израз1: оператор 11;оператор 12;.....; break;
  case   константен израз 2: оператор 21;оператор 22;.....; break;
  ..............
  case   константен израз n: операtр 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***. Default–частта в оператор 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.

4. Конструкция break

Конструкция ***break*** предизвиква завършване на итерационните концтрукции (***while***, ***do-while***, ***for***) или на конструкция ***switch***.

За правилната реализация на алгоритми чрез конструкция ***switch***, се използва ***break***. Това се гарантира, че след изпълнение на един от вариантите, останалите няма да бъдат проверявани т.е. ще бъде избран само един от възможните варианти.

Синтаксис:

break;

5. Реализация на разклонения чрез *оператор за условен израз* (*?:*)

Език С разполага с редица полезни оператори, които повишават ефективността на програмирането. Пример за това е операторът за условен израз.

Синтаксис:

операнд 1 ? операнд 2: операнд 3;

Операндите могат да са:константи, променливи, изрази. Операторът изисква 3 операнда. Действието е следното: Пресмята се операнд 1. Ако резултатът е различен от 0 (т.е. резултатът е TRUE), изчислява се резултатът от операнд 2 и неговата стойност е резултатът от оператора. Ако резултатът е от операция 1 е равен на 0 (т.е. резултат FALSE), изчислява се операнд 3 и стойността му е резултат от оператора.

Пример:

```
...
int a,b,c;
...
c=(a>0)?a:b;   /* ако a>0, на c се присвоява стойността на a, в противен случай, на c се присвоява стойността на b */
```

6. Примери за реализация на разклонени алгоритми

Пример 1:

// програма за пресмятане корените на квадратното уравнение

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



int main()
{
   int a,b,c;           // коефициенти на квадратното уравнение
   double d,x1,x2;      // дискриминанта, корени на квадратното уравнение
```

```
   printf("\n a=");
   scanf("%d",&a);
   printf(" b=");
   scanf("%d",&b);
   printf(" c=");
   scanf("%d",&c);

   if(a)                              // проверка дали уравнението е  линейно
    { d=(b*b+4*a*c);                 // проверка за отрицателна дискриминанта
      if(d<0) printf("Няма реални корени");
      else
       if(d)                         // проверка дали дискриминантана е 0
           { 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("Линейно уравнение!");
 }
```

Пример 2:

Пресмятане на стойността на функция y=f(x), според зададена стойност на параметър k

Задачата е: да се състави програма за пресмятане на функцията y като:
$y=x^2+2x+16$, при k=1
$y=\ln x+2x$, при k=2
$y=x^5+2x^3+12$, при k=3

Програмата е пример за използване на оператор ***switch***

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

main()
  {  int k;
      float x;
     double y;
     printf("Enter the value of  k 1 - 3:");
     scanf("%d",&k);
     printf("Enter the value of  x:");
     scanf ("%f",&x);
     switch (k){
                case 1: y=(x*x)+(2*x)+16;
                            printf ("the result is  y=%f",y);
                            break;

                 case 2: y=log(x)+(x*x);
                             printf ("the result is  y=%f",y);
                       break;
                 case 3:  y=pow(x,5)-2*pow(x,3)+12;
                              printf ("the result is  y=%f",y);
                        break;
   }
```

```
 return 0;
}
```

II. Задачи за изпълнение

1. Да се създаде конзолно приложение за въвеждане на две числа от клавиатурата и намиране на по-голямото от тях. Задачата да се реализира в два варианта: чрез използване на конструкция if-else и, чрез оператор за условен израз.

2. Да се създаде конзолно приложение, чрез което се определя вида на триъгълник (равностранен, равнобедрен, разностранен) по зададени три страни.

3. Да се създаде конзолно приложение, което определя вид на триъгълник по за дадени 3 страни. Да се намери лицето на триъгълника като се използва Хиронова формула. В решението на задачата да е включена проверка за коректност на входните данни.

Упътване:

Ако a,b,c са страните на триъгълник, лицето на триъгълника се намира по следната формула: $s = \sqrt{p(p-a)(p-b)(p-c)}$ *като* $p = \frac{a+b+c}{2}$

Условието a, b, c да са страни на триъгълник е сборът на две съседни страни да е по-голям от третата: $a+b>c$, $b+c>a$, $a+c>b$

4. Да се състави вариант на алгоритъм и програма за пресмятане корените на квадратно уравнение.

5. Да се състави програма за намиране стойностите на функцията y по зададен аргумент x:

y=2, при x<=0;
y=x+2, при x (0,1);
y=3, при x [1,2];
y=5-x, при x (2,3);
y=2, при x>=3;

6. Да се състави алгоритъм и програма за пресмятане лице на: триъгълник, при k=1; квадрат, при k=2; правоъгълник, при k=3; кръг, при k=4. Стойностите за параметър k, както и данните за всяка фигура да се въвеждат от клавиатурата.