Лабораторно упражнение N 5
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 с се присвоява стойността на a, в противен случай, на с 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>

cdecl 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.