Лабораторно упражнение № 4
Итерационни конструкции. Реализация на циклични алгоритми

I. Теоретична обосновка

Циклични алгоритми са тези при които има многократно повторение на група действия. Многократното повторение на група действия се нарича ***цикъл***. Конструкциите за цикли в С са част от управляващите, наричат се още **итерационни**. Предвидени са три итерационни конструкции, които реализират цикли в програмите: while, do-while, for. Тялото на цикъла се състои от един оператор. Ако алгоритъмът изисква в тялото да са повече от един оператори, те се обединяват чрез съставният оператор **{…}**.

1. Конструкция while

Чрез конструкция while се реализира цикъл с пред условие.Синтаксисът е следният:
while (условие) оператор;

Действието е следното: Проверява се условието. Ако стойността на израза е различна от 0 (TRUE), изпълнява се тялото на цикъла и отново се проверява условието за край, т.е.оператора в тялото за цикъла се изпълнява, докато условието е вярно. Когато стойността на условието стане равна на 0 (FALSE), излиза се от цикъла.

2. Конструкция за цикъл със след условие do-while

Конструкция do-while реализира цикъл със след условие. Синтаксисът е следния:
do оператор;
while (условие);

Действието е следното: Операторът в тялото на цикъла се изпълнява до тогава докато условието е вярно т.е изразът представящ условието има стойност различна от 0 (TRUE). Когато стойността на израза стане равна на 0 (FALSE), прекратява се изпълнението на цикъла.

3. Конструкция for

Итерационна конструкция for се използва предимно за реализация на цикли с известен брой повторения. По своето действие, конструкцията реализира цикъл с предусловие.

Синтаксисът е следния:
for (секция 1;секция 2;секция 3) оператор;

Секция 1 и секция 3 се състоят от един или няколко оператора, отделени със запетая. Секция 2 е условен израз.

Действието е следното: Изпълняват се операторите в секция 1. Това е т.нар. ***инициализираща секция***, която се изпълнява еднократно и с която се задават началните условия на цикъла. Изпълнението на for продължава в следната последователност: изчисляване на секция 2; изпълнение на оператора след скобите; изпълнение на секция 3. Тази последователност се изпълнява дотогава, докато стойността на израза в секция 2 има стойност различна от 0 (TRUE).

4. Примери за реализация на циклични алгоритми

4.1. Намиране на N факториел

```
// пример 1- намиране на N факториел чрез използване на оператор while
#include <iostream.h>

main()
{ int i;
  unsigned n,nf;

  cout<<" Input n=";
  cin>>n;

  nf=1;
  i=1;
  while(i<=n)
   { nf*=i;
     i++;
   }
 cout<<"\n N facturiel = "<<nf;
 return 0;
}

// пример 2: намиране на N факториел чрез използване на оператор do-while
#include <iostream.h>

main()
{ int i;
  unsigned n,nf;

  cout<<" Input n=";
  cin>>n;
  nf=1;
  i=1;
  do
   { nf*=i;
     i++;
   }  while(i<=n);
 cout<<"\n N facturiel = "<<nf;
 return 0;
}

// пример 3: намиране на N! чрез използване на оператор for

#include <iostream.h>

main()
{ int i;
  unsigned n,nf;

  cout<<" Input n=";
  cin>>n;

  nf=1;
  for(i=1;i<=n;i++) nf*=i;
  cout<<"\n N facturiel = "<<nf;
 return 0;
}
```

4.2. Намиране на средноаритметична стойност

Задачата е за намиране на средноаритметична стойност от произволен брой положителни числа, въведени от клавиатурата.

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

main ()
{ float x,sum,srst;
   int k;
   k=0;
   sum=0;
  do
    {  printf ("krai na vavejdaneto e 0\n");
        printf ("vavedi  x na broi polojitelni chisla:\n");
         scanf ("%f",&x);

        if (x>0)
          { k++;
             sum+=x;
           }
   }  while(x!=0);
  srst=sum/k;
  printf ("srst=%f",srst);
  return 0;

}
```

II. Задачи за изпълнение

1. Да се създаде конзолно приложение, което намира сумата на произволно въведени положителни числа от клавиатурата. Въвеждането на стойност 0 да прекрати понататъшното въвеждане на числа.

2. Да модифицира програмния код в задача 1, така че приложението да намира произведението на произволно въведени положителни числа от клавиатурата.

3. Да се създаде конзолно приложение, с което се извеждат на екрана стойностите от 1 до n в прав и обратен ред, като стойността на n се въвежда от клавиатурата.

4. Да се създаде конзолно приложение, което отпечатва на екрана следното:

```
1
1 2
1 2 3
1 2 3 4
…
1 2 3 4 … n
```

Стойността на n се въвежда от клавиатурата.

5. Да се създаде конзолно приложение, което отпечатва на екрана следното:

```
       0
      101
     21012
    3210123
n..3210123..n
```

Стойността на n се въвежда от клавиатурата.

6. Да се състави алгоритъм и програма за намиране на e^x в ред на Фурие. Натрупването на междинната сума да се прекрати, когато поредния член стане по-малък от 10^{-8} . Формулата за пресмятане е: $e^x=1+x/1+x^2/2!+x^3/3!+...+x^n/n!$

Да се сравни получената стойност със стойността, получена чрез използване на функция exp().

Лабораторно упражнение № 5
Дефиниране и обработка на масиви

I. Теоретична обосновка

1. Дефиниране и използване на масиви

Типовете данни в програмните езици се разделят най-общо на ***скаларни*** и ***структурирани***. Скаларни типове данни са тези, които се състоят от един елемент. В С, пример за такива данни са аритметичните типове ***int***, ***char***, ***float*** и ***double***. Структурираните типове се състоят от повече от един елемент

Масивът е структура от данни, състояща се от множество последователно наредени елементи от един и същи тип, достъпът до който се осъществява чрез името на масива поредния номер на елемента т.нар. **индекс.**

Едномерни масиви се дефинират чрез следната декларация:

име_тип име_масив[размерност];

име_тип - определя типа на елементите на масива. Може да е някой от стандартните типове или дефиниран от програмиста;

име_масив е идентификатор, определящ името на променливата от тип масив;

размерност - константен израз, който определя броя на елементите в масива и се нарича още ***граница***.

Примери:

```
int massiv[10];        // дефиниране на едномерен масив massiv от 10 цели елемента
char ch[20];           // дефиниране на едномерен масив ch от 20 символа
float a[10], b[20];    // дефиниране на два едномерни, реални масива a и b от 10 и 20
                       // елемента, съответно
```

Достъпът до даден елемент на масива е с името на масива и индекса на елемента.

Пример:

```
a[2]=0;
b[18]=1;
ch[10]='y';
```

Особеност на език С, относно масивите е, че индексите на елементите започват от 0, т.е. ако даден масив е с N елемента, първият елемент е с номер 0, а последният е с номер N-1.

Дефинирането на многомерни масиви е чрез следната декларация:

име_тип име_масив[граница1][граница2].....;

Примери:

```
float ax[10][20];      // дефиниране на двумерен масив ax от 10x20 реални елемента
int bx [10][10][5];    // дефиниране на тримерен масив bx от 10x10x5 целочислени
                       // елемента
```

При дефинирането на масиви могат да бъдат зададени начални стойности на елементите, т.е. масивите да бъдат инициализирани. Общият вид на дефиницията е следния:

име_тип име_масив [размерност]={списък стойности};

Стойностите на елементите се разделят със запетаи.

Пример:
int array[3] = {1, 3 , 5};
/* съответно стойностите на елементите са: array[0]=1; array[1]=3; array[2]=5 */

2. Примери за използване на масиви

В упражнението е представено решението на две примерни задачи, свързани с използването на масиви като структури от данни.

2.1. Търсене на минимална стойност

Задачата е да се намери позицията и стойността на минимален елемент в едномерен масив от 10 целочислени елемента.

```
#include <stdio.h>

main ()
{ int mas[10];
  int i,j,a,k, min;

 printf ("insert the number of the massive=");
 scanf ("%d",&j);

  for(a=0;a<j;a++) scanf("%d",&mas[a]);

  min=mas[0];                    // min съдържа стойността на най-малкият
елемент
  k=0;                          // k съдържа позицията на най-малкият елемент
  for (i=1; i<j;i++)
        if (mas[i]<min)
              { min=mas[i];
                k=i;
              }
  printf ("min=%d\n",min);
  printf ("poziciqta e %d\n",k);
  return 0;
}
```

2.2. Сумиране на матрици

В примерната програма е решението на задачата за въвеждане на стойностите на две матрици (a, b)и намиране на тяхната сума в резултантна матрица c.

```
#include <iostream.h>

void main()

{ const M=20;  // максималният размер на матриците е 20x10
  const N=10;
  int a[M][N], b[M][N], c[M][N];
  int i,j,m,n;      // m,n – действителен размер на матриците

  do {
```

```
        printf("m=");
       scnaf(“%d”,&m);
     } while ((m>20)||(m<2));        // стойността на m да е в интервала [3,20]
  do {
        printf("n=");
       scnaf(“%d”,&n);

     } while ((n>10)||(n<2));          // стойността на n да е в интервала [3,10]

  for(int i=0;i<m;i++)
   for(int j=0;j<n;j++)   scanf(“%d”,&a[i][j]);     // въвеждане на матрица a

  for(int i=0;i<m;i++)
    for(int j=0;j<n;j++)   scanf(“%d”,&b[i][j]);    // въвеждане на матрица b

  for(int i=0;i<3;i++)                          // сумиране на матриците
   for(int j=0;j<3;j++)   c[i][j]=a[i][j]+b[i][j];

  for(int i=0;i<3;i++)                          // извеждане на матрица c
     {for(int j=0;j<3;j++) printf(“%d  “,c[i][j]);
      printf(“\n”);;
     }
}
```

II. Задачи за изпълнение

1. Да се създаде конзолно приложение, с което се въвеждат 10 цели числа и се извеждат в прав и обратен ред.

2. Да се създаде конзолно приложение, с което се намира сумата и средно аритметично на елементите на едномерен масив от 10 реални числа с двойна точност.

3. Да се създаде конзолно приложение, с което се намира стойността и позицията на максималния елемент в едномерен масив от 10 целочислени елемента.

4. Да се създаде конзолно приложение за сумиране на две матрици. Стойностите на елементите на входните матрици да се въвеждат от клавиатурата.

5. Да се създаде конзолно приложение за въвеждане елементи на двумерен масив и намиране сумата на елементите във всеки ред.