П И И С

График на упражненията
MS Visual Studio C++ 6.0
за студентите от задочна форма на обучение, специалности ССС, ТС, ВиК и ХТС,
първи курс
за учебната 2015/2016 година

Упр. 1 – 16.09.2015 (сряда) от 13 до 17 часа (4 часа)

Тема 1: Процес на програмиране **CPP –> OBJ –> EXE**. Програмна среда – съдържание. Първа задача: **S=a+b (int)** – обяснения, библиотека **iostream.h**, оператори **cin, cout.** Вариации на задачата (изваждане, умножение, деление на **a** и **b**).

Тема 2: Типове данни – **int, float, double, char**. Деклариране и инициализиране на променливи. Деклариране на константи. Библиотека **math.h** и функции **sqrt(x), pow(x,y)**. Задача: Изчисляване на периметър и лице на триъгълник по зададени координати на върховете (2D). Програма с **void main ()**

Тема 3: Видове операции – **бинарни, унарни, тернарни; леви, десни**. Приоритет. Аритметични операции. Релационни операции (резултат **0, 1**). Логически операции (резултат **0, 1**). Оператор за присвояване, модификации, **i++, i--.** Задачи – разстояние между 2 точки в 3D, водна смес (самостоятелно).

Тема 4: Съставен оператор. Условен оператор **if** – проста форма, разширена форма. Библиотека **math.h** – всички функции. Локални и глобални променливи. Оператор **goto**. Задача: квадратно уравнение $\mathbf{a\ x^2 + b\ x + c = 0}$; тестване поне с 3 комплекта данни.

Упр. 2 – 17.09.2015 (четвъртък) от 8 до 12 часа (4 часа)

Тема 5: Оператори **switch, break**. Пример: **S=a+b (+ ; – ; * ; /)**. Пример с дни от седмицата. Оператор за цикъл **for**.

Задача: $S = \sum_{i=1}^{100} i \quad ; \quad S = \sum_{i=1}^{100} \sqrt{i} \quad ; \quad S = \sum_{i=1}^{10} i^2$

Масиви – определение, едномерни, двумерни, деклариране, посочване на елементи.

Тема 6: Задача: $P = N! = \prod_{i=1}^{N} i \; ; P = \prod_{i=M}^{N} i$

Едномерни масиви: $S = \sum_{i=1}^{N} x_i \; ; \; P = \prod_{i=1}^{N} x_i \; ; x_i \neq 0,$

Сума, произведение на ненулеви елементи; средно аритметично на положителни елементи - всичко обединено в една голяма задача.

Тема 7: Оператори **while** и **do – while**. Едномерен масив – продължение: max(x), min(x); No на максималния и минималния елемент. Сортировка – метод на мехурчето. Примери.

Упр. 3 – 17.09.2015 (четвъртък) от 16 до 17 часа (1 час)

Тема 8: Указатели – определение, операции. Указатели и масиви. Програми.

Упр. 4 – 19.09.2015 (събота) от 10 до 12 часа (2 часа)

Тема 9: Указатели към символни променливи и символни масиви.
Структури – кратко запознаване.

Тема 10, 11: Функции, предаване на параметри. Формални и фактически параметри.

Упр. 5 – 20.09.2015 (неделя) от 13 до 17 часа (4 часа)

Тема 12: Функции – заключително упражнение.
Файлове. Функции за четене и запис във файл.
Курсови задачи.
Заверка.

Тема 1:

Процес на програмиране **CPP –> OBJ –> EXE**. Програмна среда – съдържание. Първа задача: **S=a+b (int)** – обяснения, библиотека **iostream.h**, оператори **cin, cout.** Вариации на задачата (изваждане, умножение, деление на **a** и **b**).

o o o o o o o o o o o o o o o o o o o o o o o o o o o o o o o

```
// Първа програма на C++; MS Visual Studio C++ 6.0
#include <iostream.h>
int a,b,S;

int main () {
  cout<<"a="; cin>>a;
  cout<<"b="; cin>>b;
  S=a+b; // Изчисления; после промяна с –,*,/
  cout<<"S="<<S<<endl;
  return 0;
}
```

Step 1: File > New (Project) > Win32 Console Application (Project Name, Location, An Empty Project)

Step 2: File > New (File) > C++ Source file (Filename; √ Add to project)

```
// Първа програма в зала 515 – Visual Studio 2010
// Компилиране: F7   Стартиране: Ctrl + F5
#include <stdafx.h>
#include <iostream>

int a,b,S;

int main () {
  std::cout<<"a="; std::cin>>a;
  std::cout<<"b="; std::cin>>b;
  S=a+b; // Изчисления; после промяна с –,*,/
  std::cout<<"S="<<S<<'\n';
  return 0;
}
```

или

```
// Първа програма в зала 515 – Visual Studio 2010
// Компилиране: F7   Стартиране: Ctrl + F5
#include <stdafx.h>
#include <iostream>
using namespace std;

int a, b, S;

int main () {
  cout<<"a="; cin>>a;
  cout<<"b="; cin>>b;
  S=a+b; // Изчисления; после промяна с –,*,/
  cout<<"S="<<S<<'\n';
  return 0;
}
```

Тема 2:

Типове данни – **int, float, double, char**. Деклариране и инициализиране на променливи. Деклариране на константи. Библиотека **math.h** и функции **sqrt(x), pow(x,y)**. Задача: Изчисляване на периметър и лице на триъгълник по зададени координати на върховете (2D). Програма с **void main ()**

o o o o o o o o o o o o o o o o o o o o o o o o o o o o o o

```
#include <iostream.h>
#include <math.h>
float X1,Y1,X2,Y2,X3,Y3,a,b,c,p,P,S;

void main () {
  cout<<"X1="; cin>>X1;
  cout<<"Y1="; cin>>Y1;
  cout<<"X2="; cin>>X2;
  cout<<"Y2="; cin>>Y2;
  cout<<"X3="; cin>>X3;
  cout<<"Y3="; cin>>Y3;
  a=sqrt(pow(X2-X1,2)+ pow(Y2-Y1,2));
  b=sqrt(pow(X2-X3,2)+ pow(Y2-Y3,2));
  c=sqrt(pow(X3-X1,2)+ pow(Y3-Y1,2));
  P=a+b+c;
  p=P/2;
  S=sqrt(p*(p-a)*(p-b)*(p-c));
  cout<<"P="<<P<<endl;
  cout<<"S="<<S<<endl;
}
```

Тема 3:

Видове операции – **бинарни, унарни, тернарни; леви, десни**. Приоритет.
Аритметични операции. Релационни операции (резултат **0, 1**). Логически операции (резултат **0, 1**). Оператор за присвояване, модификации, **i++, i--.** Задача – разстояние между 2 точки в 3D, водна смес (самостоятелно).

o o o o o o o o o o o o o o o o o o o o o o o o o o o o o o

```
// Задача 1 – самостоятелно; разстояние между 2 точки в пространството
#include <iostream.h>
#include <math.h>
float X1,Y1,Z1,X2,Y2,Z2,d;

void main () {
  cout<<"X1="; cin>>X1;
  cout<<"Y1="; cin>>Y1;
  cout<<"Z1="; cin>>Z1;
  cout<<"X2="; cin>>X2;
  cout<<"Y2="; cin>>Y2;
  cout<<"Z2="; cin>>Z2;
  d=sqrt(pow(X2-X1,2)+ pow(Y2-Y1,2)+ pow(Z2-Z1,2));
  cout<<"d="<<d<<endl;
}
```

```
// Задача 2 – самостоятелно; водна смес
#include <iostream.h>
float V1,T1, V2,T2, V,T;

void main () {
  cout<<"V1="; cin>>V1;
  cout<<"T1="; cin>>T1;
  cout<<"V2="; cin>>V2;
  cout<<"T2="; cin>>T2;
  V=V1+V2;
  T=(V1*T1+V2*T2)/V;
  cout<<"V="<<V<<endl;
  cout<<"T="<<T<<endl;
}
```

Тема 4:

Съставен оператор. Условен оператор **if** – проста форма, разширена форма. Библиотека **math.h** – всички функции. Локални и глобални променливи. Оператор **goto**. Задача: квадратно уравнение $\mathbf{a\ x^2 + b\ x + c = 0}$; тестване поне с 3 комплекта данни.

o o o o o o o o o o o o o o o o o o o o o o o o o o o o o o o o o o o

// Изчисляване на y като функция от x.

Y=f(x)= | за x>a $y = (x^2 + c)/2$

| за x<=a $y = (x^4 + c)/4$

```
#include <iostream.h>
#include <math.h>
float x,a,y;

void main () {
  cout<<"x="; cin>>x;
  cout<<"a="; cin>>a;
  cout<<"c="; cin>>c;
  if (x>a) y=(pow(x,2)+1)/2;
  else y=(pow(x,4)+1)/4;
  cout<<”y=”<<y<<endl;
}
```

// Решаване на квадратно уравнение

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

float a,b,c,d,x1,x2;

void main () {
  cout<<"a="; cin>>a;
  cout<<"b="; cin>>b;
  cout<<"c="; cin>>c;
  if (a==0) cout << “Не е квадратно уравнение.”
  else {
     d=b*b-4*a*c;
     if (d>0) {
        x1=(-b+sqrt(d))/(2*a);
        x2=(-b-sqrt(d))/2/a;
        cout<<"x1="<<x1<<endl;
        cout<<"x2="<<x2<<endl;
     }
     else
        if (d==0) {
           x1=x2=-b/(2*a);
           cout<<"x1=x2="<<x1<<endl;
        }
        else cout<<”Няма реални корени.”;
  }
}
```

Тема 5:

Оператори **switch, break**. Пример: **S=a+b (+ ; – ; * ; /**). Пример с дни от седмицата.
Оператор за цикъл **for**.
Задачи: $S = \sum_{i=1}^{100} i$; $S = \sum_{i=1}^{100} \sqrt{i}$
Масиви – определение, едномерни, двумерни, деклариране, посочване на елементи.

o o o o o o o o o o o o o o o o o o o o o o o o o o o o o o o o o o o

```
// Програма 1 – Малък калкулатор
#include <iostream.h>
float a,b,x; char c;

void main () {
  cout<<"a="; cin>>a;   // въвеждат се числовите стойности
  cout<<"b="; cin>>b;
et1: cout<<"Операция: "; cin>>c; // въвежда се операцията

  switch (c) {
     case '+': x=a+b; break;
     case '-': x=a-b; break;
     case '*': x=a*b; break;
     case '/': x=a/b; break;
     default: cout<<"Грешка!\n"; goto et1; // при зададен грешен символ скок към et1
  }
  cout<<"Резултат: "<<a<<c<<b<<"="<<x<<endl;
}

// Програма 2 – Дни от седмицата
#include <iostream.h>
int a;

void main () {
et2:  cout<<"Number: "; cin>>a;  // въвежда се число
  switch (a) {
     case 1: cout<<”Monday”; break;
     case 2: cout<<”Tuesday”; break;
     case 3: cout<<”Wednesday”; break;
     case 4: cout<<”Thursday”; break;
     case 5: cout<<”Friday”; break;
     case 6: cout<<”Saturday”; break;
     case 7: cout<<”Sunday”; break;
     default: cout<<"Error!\n"; goto et2; // при грешка – връщане към началото
  }
  if (a>=1 && a<=5) cout<<” is work day.”
  else cout<<” is day of rest.”
}
```

Оператор с управляваща променлива for:
for (инициализация; условие; промяна) оператор;

```
// Програма 3 – сума
#include <iostream.h>
#include <math.h>

int i; float S;

void main () {
    S=0;
    for (i=1; i<=100;i++) S=S+sqrt(i);
    cout<<”S=”<<S<<endl;
}
```

Масиви:
А) едномерни
float x(100); // елементи от x[0] до x[99]

Б) двумерни

В) тримерни

Тема 6:

Задача: $P = N! = \prod_{i=1}^{N} i$

Едномерни масиви: $S = \sum_{i=1}^{N} x_i$; $P = \prod_{i=1}^{N} x_i$; $x_i \neq 0$,

Едномерен масив – сума, произведение, средно аритметично на положителни елементи; max(x), min(x); No на максималния и минималния елемент. Всичко обединено в една голяма задача.

o o o o o o o o o o o o o o o o o o o o o o o o o o o o o o o o o o o

```
// Програма – факториел
#include <iostream.h>
#include <math.h>
int i,N;
int P;  // float

void main () {
    cout<<”N=”; cin>>N;
    P=1;
    for (i=1; i<=N;i++) P=P*i;  // вариант: for (i=1, P=1; i<=N; P*=i, i++);
    cout<<”P=”<<N<<”!=”<<P<<endl;
}
```

Вариация на оператор for: for (i=1,P=1;i<=N;P*=i,i++);

```
// Задача 2: Даден е едномерен масив A(N) с максимум 100 цели елемента.
// а) Да се намери сумата от всички елементи;
// б) да се намери произведението от всички ненулеви елементи;
// в) да се изведат всички четни елементи;
// г) да се намери средното аритметично на всички отрицателни елементи;
// д) да се намери най-големият елемент в масива и неговия индекс;
// е) да се намери най-малкият елемент в масива и неговия индекс;
// ж) копирайте масива в нов масив B(N), сортирайте масива и изведете елементите във
възходящ ред.
// з) намерете скаларното произведение на масивите A и B.

#include <iostream.h>
int A[100];
int N,i;
void  main () {
// въвеждане на N в интервала [2,100]
  do {
    cout << "Въведете брой елементи N="; cin >> N;
  }
  while (N<2 || N>100);

// въвеждане на едномерния масив от цели елементи
  for (i=0; i<N; i++) {
    cout<<"A["<<i<<"]="; cin >> A[i];
  }

// а) Да се намери сумата от всички елементи;
```

```
int S=0;
  for (i=0; i<N; i++) S += A[i];
  cout "Сумата на всички елементи е S=" << S << endl;

// б) да се намери произведението от всички ненулеви елементи;
int P=1;
  for (i=0; i<N; i++)
    if (A[i] != 0) P *= A[i];
  cout "Произведението от всички ненулеви елементи е P=" << P << endl;

// в) да се изведат всички четни елементи;
  for (i=0; i<N; i++)
    if (A[i] % 2 == 0) cout << A[i] << '\t';
  cout << endl;

// г) да се намери средното аритметично на всички отрицателни елементи;
float SA;
int Sm=0, Nm=0;
  for (i=0; i<N; i++)
    if (A[i] < 0) {
      Sm += A[i]; Nm++;
    }
  if (Nm>0) {
    SA= 1. * Sm / Nm; // 1. е нужно, за да се получи реален, а не цял резултат от деленето
    cout << "Ср. аритметично на всички отрицателни елементи е SA=" << SA << endl;
  }
  else cout << " Няма отрицателни елементи в масива\n";

// следва в упражнение 7
```

Тема 7:
(продължение)

```
// д) да се намери най-големият елемент (максимумът) в масива и неговия индекс;
int Max, imax;
  Max=A[0]; imax=0;
  for (int i=1; i<N; i++)
    if (A[i] > Max) {Max = A[i]; imax=i; }
  cout << "Най-големият елемент в масива е Max=A[" <<imax<<"]="<< Max << endl;

// е) да се намери най-малкият елемент (минимумът) в масива и неговия индекс;
int Min, imin;
  Min=A[0]; imin=0;
  for (int i=1; i<N; i++)
    if (A[i] < Min]) {Min = A[i]; imin=i; }
  cout << "Най-малкият елемент в масива е Min=A[" <<imin<<"]="<< Min << endl;

// ж) копирайте масива в нов масив B(N), сортирайте масива и изведете елементите във
възходящ ред. (към Тема 7)

int j, t, B[100];
  for (i=1;  i < N; i++) B[i]=A[i];
  for (i=1;  i < N; i++)
    for (j=1; j <= N - i; j++)
      if (B[j-1) > B[j]) {
        t=B[j-1]; B[j-1]=B[j]; B[j]=t;
      }
  cout << "Подреденият във възходящ ред масив е:\n";
  for (i=0; i<N; i++)
    cout<<"B["<<i<<"]=" << B[i] << endl;

// з) намерете скаларното произведение на масивите A и B.
int Sp=0;
  for (i=0; i<N; i++) Sp+=A[i]*B[i];
    cout<<"скаларното произведение на двата масива е Sp="<<Sp<<endl;

}
```

Тема 8:

Указатели – определение, операции. Указатели и масиви. Програми.
Раздаване на курсови задачи

o o o o o o o o o o o o o o o o o o o o o o o o o o o o o o o o o o o

```
// Задача 1 за указатели към масиви
#include <iostream.h>
float x[10], *px;

int N, i;

void main() {
et1: cout<<"N="; cin>>N;
  if (N<2 || N>10) goto et1;

for (i=0; i<N; i++) {
  cout<<"x["<<i<<"]=";  cin>>x[i];

  px=&x[i]; *px=*px+1;
  cout<<"px="<<px<<'\n';
  cout<<"*px="<<*px<<'\n';
}
}
```

Тема 9:

Указатели към символни променливи и масиви.
Структури.

o o o o o o o o o o o o o o o o o o o o o o o o o o o o o o o o o o o

```
// Задача 2 за указатели към числови масиви и указатели към символни масиви
#include <iostream.h>
float a[4], *pa;

void main() {
  pa=a; *pa=10;
  pa++; *pa=20;
  pa=&a[2]; *pa=30;
  pa=a+3; *pa=40;

  for (int i=0; i<4; i++)
    cout<<"a["<<i<<"]="<<a[i]<<endl;

// работа с масив от указатели към два символни низа
  char *pt[]={"Sunday", "Monday"};
  cout<<"pt[0]="<< pt[0] <<endl;      // извежда думата Sunday
  cout<<"*pt[0]="<< *pt[0] <<endl;  // извежда символ S
  cout<<"pt[1]="<< pt[1] <<endl;      // извежда думата Monday
  cout<<"*pt[1]="<< *pt[1] <<endl;  // извежда символ M

}
```

Работа със структури

```
// Примерна програма за работа със структури
// Дадени са N точки в равнината, зададени с техните координати, като те образуват
изпъкнал N-ъгълник.
// Да се изчисли периметъра на N-ъгълника.

#include <iostream.h>
#include <math.h>

const Nmax=100; // Максимално допустим брой точки

// структура за точка
struct point_str {
  double x,y;
};


// деклариране на масив от структурни данни за точки
point_str P[Nmax];

double L;
```

```
void main () {
// въвеждане на N в интервала [2,100]
  do {
    cout << "Въведете брой точки N="; cin >> N;
  }
  while (N<2 || N>Nmax);

// въвеждане на координатите на точките
  for (i=0; i<N; i++) {
    cout<<"X["<<i<<"]="; cin >> P[i].x;
    cout<<"Y["<<i<<"]="; cin >> P[i].y;
  }

double A[Nmax]; // дължини на страните на N-ъгълника

  L=0;

  for (i=0; i<N; i++) {
    if (i < N-1) // i-тата страна свързва точки i и i+1
      A[i]=sqrt(pow(P[i].x-P[i+1].x,2) + pow(P[i].y-P[i+1].y,2));
    else // при i=N-1 (последна точка) i-тата страна свързва точки i и 0
      A[i]=sqrt(pow(P[i].x-P[0].x,2) + pow(P[i].y-P[0].y,2));
    L += A[i];
  }
  cout << "Обиколката на многоъгълника е L=" << L << endl;
}
```

Тема 10:

ИЗПОЛЗВАНЕ НА ФУНКЦИИ В C++
(част 1)

Функциите са основни структурни единици в езика C++. Те представляват обособени самостоятелни части от програмата, т.е. нещо като подпрограми. Най-често във функции се обособява описанието на задачи, които често се изпълняват в различни програми, например въвеждане на елементи на масив, извеждане на елементи на масив, търсене на минимум или максимум, изчисляване на разстояние между точки и т.н.

Особеното за тях е, че те могат да връщат резултат към извикващата ги част от програмата. Това става с помощта на оператора **return**.

Примерно описание на функция за изчисляване на разстояние между две точки е:

```
float dist (float x1, float y1, float z1, float x2, float y2, float z2) {
float d;
  d=sqrt(pow(x2-x1,2)+pow(y2-y1,2)+pow(z2-z1,2));
  return d;
}
```

Името на функцията е **dist**. Типът на резултата, връщан от нея, е **float**. В този пример **x1, y1, z1, x2, y2, z2** са **формални параметри**, които се предават от извикващата част към функцията. **d** е локална променлива за тази функция и принадлежи само към нея. Това означава, че тя има смисъл само в тази функция и не може да бъде извиквана или използвана в друга част от програмата. С помощта на оператора **return d;** стойността на **d** се връща към извикващата част от програмата. Функциите могат да използват и т.н. глобални променливи, принадлежащи на програмата и дефинирани преди самата функция.

Функцията от горния пример може да се извика по следния начин за изчисляване на разстояние между две точки в равнината:

```
   z[k]=dist(x[i],y[i],0,x[j],y[j],0);
```

Към функцията се предават **фактически стойности** на константи, реални променливи или на елементи от реален масив. Изчисленият от функцията резултат се присвоява на реална променлива или на елемент от масив, както е в конкретния случай.

Възможно е функцията да има параметри от различен тип, а също и да не връща резултат, а само да изпълнява определена последователност от действия. В такъв случай функцията е от тип **void**:

```
void write_element(char Name, int i, float x) {
  cout <<Name<<"["<<i<<"]="<<x;
}
```

Извикването на тази функция става с:

```
write_element('A',i,A[i]);
```

В този случай трябва да се внимава броят и типът на данните в извикващата част да съответства на броя и типа на параметрите, описани в самата функция. Механизмът на работа на функцията е такъв, че нормално към функцията се предават копия на стойностите на параметрите. По тази причина промяната на стойностите на параметрите вътре в самата функция не се отразява на стойностите на съответните данни в извикващата част.

Най-често извикващата част предава данни на функцията, а функцията може да върне само една стойност чрез **return**. В някои случаи може да се наложи функцията да върне повече от една стойности или да промени стойностите на параметрите. Това може да стане чрез използването на указатели.

Типичен пример е функция, която трябва да размени стойностите на две променливи от еднакъв тип, например реални:

void swap_float(float *pa, float *pb){ //pa и pb са имена на указатели към две реални променливи
float c;
c=*pa; *pa=*pb; *pb=c;
}

Тази функция трябва да бъде извикана по следния начин:
swap_float(&a, &b);

По този начин към функцията се предават адресите на променливите **a** и **b**, които представляват стойности на указателите **pa** и **pb**. Размяната на стойностите на адресите, към които сочат указателите представлява размяна на самите стойности на променливите **a** и **b.**

Друг начин да се изпълни същата задача е чрез псевдоними:
```
void  swap_float2(float &a, float &b) {
float c;
   c=a; a=b; b=c;
}
```

Тази функция трябва да бъде извикана по следния начин:
swap_float2(a, b);

Полезни функции:

1. Функция, която въвежда цяла променлива в определен диапазон:

int Input_N(char Prompt[40], int Nmin, int Nmax){ //Prompt е подсказващ текст при въвеждането
```
int N;
 do {
   cout <<Prompt; cin >>N;
 }
 while ((N<Nmin) || (N>Nmax));
 return N;
}
```

Функцията може да се извика по следния начин:

```
 M=Input_N("Въведете брой редове M=", 1, 10);
 N=Input_N("Въведете брой стълбове N=", 1, 10);
```

или

```
 N=Input_N("Въведете брой точки N=", 2, 100);
```

2. Функция, която въвежда реален едномерен масив от клавиатурата:

```
const int Nmax=50;
void Input_Arr(char Name[10], int N, float X[Nmax]){
int i;
 for (i=0; i<=N-1; i++) {
   cout <<Name<<"["<<i<<"]=";
   cin >>X[i];
 }
}
```

Когато на функцията се предава масив, чиито стойности после искаме да бъдат върнати в извикващата част от програмата, както е в посочения пример, не е необходимо масивът да се предава с указател, защото това се реализира вътрешно от езика. В този случай стойностите, които са въвеждат в масива по време на изпълнение на функцията, на практика се записват по адресите на първоначалния масив, който се предава към функцията. Счита се, че името на масива е развнозначно на указателя към масива.

Функцията може да се извика по следния начин:

```
 Input_Arr("X", N, X);
```

или

```
 Input_Arr("Y", M, Y);
```

3. Функция, която извежда на екрана реален едномерен масив:

```
const int Nmax=50;
void Write_Arr(char Name[10], int N, float X[Nmax]){
int i;
 for (i=0; i<=N-1; i++)
   cout <<Name<<"["<<i<<"]="<<X[i];
}
```

Функцията може да се извика по следния начин:

```
 Write_Arr("X", N, X);
```

или

```
 Write_Arr("Y", M, Y);
```

4. Функция, която въвежда реален двумерен масив от клавиатурата:

```
const int Nmax=50;
void Input_Matrix(char Name[10], int M, int N, float X[Nmax][Nmax]){
int i, j;
 for (i=0; i<=M-1; i++)
   for (j=0; j<=N-1; j++){
     cout <<Name<<"["<<i<<','<<j<<"]=";
     cin >>X[i][j];
```

```
   }
}
```

Функцията може да се извика по следния начин:

```
 Input_Matrix("A", M, N, A);
```

или

```
 Input_Matrix("B", N, N, B);
```

5. Функция, която извежда на екрана реален двумерен масив:

```
const int Nmax=50;
void Write_Matrix(char Name[10], int M, int N, float X[Nmax][Nmax]){
int i, j;
 for (i=0; i<=M-1; i++)
   for (j=0; j<=N-1; j++)
     cout <<Name<<"["<<i<<','<<j<<"]="<<X[i][j];
}
```

Функцията може да се извика по следния начин:

```
  Write_Matrix("A", M, N, A);
```

или

```
  Write_Matrix("B", N, N, B);
```

6. Функция, която сортира във възходящ ред реален едномерен масив:

```
const int Nmax=50;
void Sort_Arr(int N, float X[Nmax]){
int i, j;
 for (i=1; i<=N-1; i++)
   for (j=0; j<=N-i-1; j++)
     if (X[j] > X[j+1]) swap_float(&X[j], &X[j+1]);
}
```

Интересното в този случай е, че тази функция използва друга функция, описана по-горе, за размяна на две реални променливи.

Функцията може да се извика по следния начин:

```
    Sort_Arr(N, Z);
```

Резултатът от извикването на тази функция е подреждането във възходящ ред на масива Z, който се предава от извикващата част на програмата към функцията и който се подрежда по време на изпълнение на тази функция.

7. Функция, която търси максимум на реален едномерен масив:

```
const int Nmax=50;
float Max_Arr(int N, float X[Nmax]){
int i;
float Max;
 Max=X[0];
 for (i=1; i<=N-1; i++)
    if (Max<X[i]) Max=X[i];
 return Max;
```

```
}
```

Аналогично може да се създаде функция, която да търси минимум на масив.
Функцията може да се извика по следния начин:

```
MaxX=Max_Arr(N, X);
```

8. Функция, която изчислява сумата на елементите на реален едномерен масив:

```
const int Nmax=50;
float Sum_Arr(int N, float X[Nmax]){
int i;
float Sum=0;
 for (i=0; i<=N-1; i++)  Sum += X[i];
 return Sum;
}
```

Функцията може да се извика по следния начин:

```
  SumX=Sum_Arr(N, X);
```

Тема 11:

ИЗПОЛЗВАНЕ НА ФУНКЦИИ В C++
(част 2)

```
// Дадени са N точки в равнината с техните координати X(N) и Y(N), като N <=100.
// Да се намери и изведе дължината на контура, ако точките са свързани последователно от първата към последната.
// Да се използват функции

#include <iostream.h>
#include <math.h>

const int Nmax=100;
float X[Nmax], Y[Nmax];
int N;


// Функция за въвеждане на цяла променлива в зададен интервал [min, max]
int Input_M(char text[30], int min, int max) {
int m;
  do {
    cout << text; cin >> m;
  }
  while (m<min || m>max);
  return m;
}

// Функция за въвеждане на координати на точки
void Input_Points(int N, float X[Nmax], float Y[Nmax]) {
  for (int i=0; i<N; i++) {
      cout<<name<<"X["<<i<<"]="; cin >> X[i];
      cout<<name<<"Y["<<i<<"]="; cin >> Y[i];
    }
}

// Функция, която търси разстояние между две точки
float dist(float X1, float Y1, float X2, float Y2) {
float d;
  d=sqrt(pow(X2-X1,2)+pow(Y2-Y1,2));
  return d;
}

// Функция, която търси сума на елементи в масив
float Sum_Array(int M, float A[Nmax]) {
float S;
  for (int i=0; i<M; i++) S+=A[i];
  return S;
}

// Главна функция
void main () {
```

```
  M=Input_M("Въведете брой точки M=",2,Nmax);
  Input_Points(M,X,Y);

float D[Nmax], S;
  for (int i=0; i<N-1; i++) D[i]=dist(X[i],Y[i], X[i+1],Y[i+1]);
  S=Sum_Array(N-1,D);
  cout<<"Дължината на контура е S="<<S<<endl;
}
```

Тема 12:

Работа с файлове

o o o o o o o o o o o o o o o o o o o o o o o o o o o o o o o o

```
// Дадени са N на брой точки в равнината, зададени чрез техните x и y координати, и
записани във файл.
// Всеки 3 последователни точки образуват триъгълник.
// Да се изчислят лицата на триъгълниците чрез векторно произведение.

#include <fstream.h>
#include <math.h>
const Nmax=100;

// Функцията чете от файл данни за точките.
// Функцията връща на извикващата част от програмата броя точки M (чрез псевдоним)
// и координатите на точките
void Load_Points(char fname[30], int &M, float X[Nmax], float Y[Nmax]) {
  ifstream fin(fname,ios::in);  // отваряне на файла за четене
  fin>>M;                       // четене на брой точки
  for (int i=0; i<M; i++)
    fin>>X[i]>>Y[i];            // четене на X и Y координати за всяка точка
  fin.close();                  // затваряне на файла
}

// Функцията извежда данните за точките на екрана
void Write_Points(char text[30], int M, float X[Nmax], float Y[Nmax]) {
  cout<<text<<endl;
  for (int i=0; i<M; i++)
     cout<<X[i]<<'\t'<<Y[i]<<endl;
}

// Функцията изчислява лицето на триъгълника по зададени координати на точките
float Area_vector(float x1, float y1, float x2, float y2, float x3, float y3) {
   float s=((x2-x1)*(y3-y1)-(x3-x1)*(y2-y1))/2;
   return fabs(s); // fabs() - функцията връща абсолютната стойност на реален параметър
}

float X[Nmax], Y[Nmax], S[Nmax-2];
int N;
char filename[30];

void main () {
  cout<<"filename="; cin>>filename;
// d:\points.txt - име на файла, където са записани данните
  Load_Points(filename, N, X, Y);   // чете данните за точките от файла
  Write_Points("Points:",N, X, Y);  // извежда данните за точките на екрана
  cout<<"Results:\n";
  for (int i=0; i<N-2; i++) {
     S[i]=Area_vector(X[i],Y[i],X[i+1],Y[i+1],X[i+2],Y[i+2]); // изчислява лицето на i-тия
триъгълник
```

```
    cout<<"S["<<i<<"]="<<S[i]<<endl;
  }
}
```

==========

Файл с входни данни:
d:\points.txt

5
1 2
3 2
3 6
0 7
1 6

Съдържание на файла:
ред 1: брой точки
ред 2 и т.н.: x и y координати на всяка точка