# Семинар 2. Модульное программирование. Подпрограммы и модули. Параметры массивы

# Передача параметров массивов в подпрограммы.

При решении многих задач для хранения и обработки данных используются массивы. Существуют приемы, позволяющие осуществлять различную обработку массивов. Эти приемы реализуют универсальные алгоритмы, которые подходят для широкого круга задач, отличающихся только типами и размерами обрабатываемых массивов. Многие из таких алгоритмов целесообразно оформить в виде функции.

Массивы можно использовать в функции двояко:

* их можно описать в теле функции;
* массивы могут быть аргументами (параметрами функции).

В силу специфики организации массивов в С\С++, массивы передаются в подпрограмму как **параметры переменные**, однако, без использования ссылок так как имя массива является его **адресом** в оперативной или динамической памяти.

В С/С++ отсутствует контроль размерности массива по первому индексу при передачи этих массивов в качестве параметров!

а)**intx[5] ⇔intx[]-** размерность проверяться не будет

б)**inty[4][8] ⇔inty[][8]** – будет проверяться размерность массива по второму индексу

В зависимости от способа распределения памяти под одномерный массив, существует несколько способов передачи параметра-массива в подпрограмму.

**1.3.1Передача в подпрограмму одномерного статического массива**

В данном случае, в качестве формального параметра подпрограммы необходимо только указать, что данная подпрограмма принимает массив элементов некоторого типа. Размерность массива указывать не обязательно. Это вызвано тем, что С\С++ проверяет соответствие фактических и формальных параметров подпрограммы только по порядку следования и по типу, причем понятие типа при этом не включает конкретной размерности массива. Из этого следует, что помимо самого массива, в качестве формального параметра необходимо передавать его размер.

**Пример 1.** Передача в подпрограмму статического массива

Написать программу построчной печати элементов массива, которая на вход принимает массив целых чисел и размер массива, а в качестве результата выводит на экран массив по k элементов. Размер массива и число k вводятся с клавиатуры.

Анализ задачи показал, что ее можно декомпозировать на следующие подзадачи.

* Формирование массива
* Вывод массива в соответствие с требованиями.

 Для обеих задач целесообразно использовать поэлементную обработку массива.

Для формирования и вывода массива разработаем подпрограммы процедуры.

**Структурная схема программы.**

**Схемы алгоритмов**

**Текст программы**

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

#include<time.h>

**void form\_arr(int a[],int n,int xn,int xk)**

{

for(int i=0;i<n;i++)

a[i]=rand()%(xk-xn+1)+xn;

}

**void print\_arr(int a[],int n,int k)**

{

for(int i=0;i<n;i++)

printf("%4d%c",a[i],((i+1)%k==0)?'\n':' ');

printf("\n");

}

int**main**()

{

int arr[100];

int n,k,a,b;

puts("Input size of arrayБ=100");

scanf("%d",&n);

srand((unsigned)time(NULL));

puts("Input begin-end diapason for rand");

scanf("%d %d",&a,&b);

form\_arr(arr,n,a,b);

printf("Input kol elem k < %3d for print\n",n);

scanf("%d",&k);

print\_arr(arr,n,k);

return0;

}

**Пример работы программы**

**1.3.2 Передача в подпрограмму многомерного статического массива**

Впроцедурном C\С++ отсутствует контроль размерности массива по первому индексу при передачи этих массивов в качестве параметров, однако, при использовании многомерного статического массива, компилятор будет проверять размерность массива по второму индексу и более индексу

а) int x[5] [10] ⇔int x[][10]- размерность по второму индексу проверяется

б) int y[4][8][5] ⇔int y[][8][5] – размерность по второмуи третьему индексам проверяется

**Пример:**

void summa(**const float x[][3], float y[],int n**)

{ inti,j;

 for(i=0;i<n;i++)

 for(y[i]=0,j=0;j<3;j++) y[i]=x[i][j];

}

floata[5][3],b[5];

………

Вызов: **summa(a,b,5);**

Рассмотрим передачу статического двумерного массива на примере удаления из матрицы указанной строки и столбца.

**Пример 2.** Написать программу удаления из матрицы l строки и k столбца с использованием подпрограмм.

Декомпозируем задачу на следующие подзадачи, которые реализуем в виде процедур.

* Формирование матрицы.
* Печать матрицы исходной матрицы.
* Удаление указанной строки.
* Удаление указанного столбца.
* Печать полученной матрицы.

Такая декомпозиция позволит более гибко применять удаление строк и столбцов, так как можно удалять не только столбец и строку одновременно, но и только строку или только столбец.

Для формирования и печати матрицы целесообразно использовать поэлементную обработку матрицы. Для удаления строк и столбцов – переформирование матрицы с изменением ее размеров.

В качестве подпрограмм-процедур реализуем все подзадачи.

**Структурная схема**

****

**Текст программы**

#include<stdio.h>

#include<math.h>

#include<time.h>

#include<conio.h>

#include<stdlib.h>

**// Процедура формирования матрицы**

**void formmatr(int a[][10],int n,int m)**

{ int i,j;

for(i=0;i<n;i++)

for(j=0;j<m;j++)

 a[i][j]=rand()/1000;

}

**// Процедура печати матрицы**

**void printmatr(int a[][10],int n,int m)**

{ int i,j;

for(i=0;i<n;i++)

 { for(j=0;j<m;j++)

 printf("%4d",a[i][j]);

 printf("\n");

 }

}

**// Процедура удаления l строки матрицы**

**void delrow(int a[][10],int& n,int m,int l)**

{int i,j;

for(i=l;i<n-1;i++)

for(j=0;j<m;j++)

 a[i][j]=a[i+1][j];

for(j=0;j<m;j++)

 a[n-1][j]=0;

 n=n-1;

}

**// Процедура удаления k столбца матрицы**

**void delcow(int a[][10],int n,int &m,int k)**

{int i,j;

for(j=k;j<m-1;j++)

for(i=0;i<n;i++)

 a[i][j]=a[i][j+1];

for(i=0;i<n;i++)

 a[i][m-1]=0;

 m=m-1;

}

int main()

{ int matr[10][10];

intn,m,l,k;

do**// контроль ввода размеров статической матрицы**

{

puts("Input n,m<=10");

 scanf("%d %d",&n,&m);

 }while((n<2)||(n>10)||(m<2)||(m>10));

 srand( (unsigned)time( NULL ));

**formmatr(matr,n,m);**

 puts("Isxodnaya Matrica");

**printmatr(matr,n,m);**

do**// контроль ввода номеров удаляемых строки и столбца**

{

 printf("Input l<= %5d k<= %5d for delete\n",n,m);

 scanf("%d %d",&l,&k);

} while((l>n)||(l<1)||(k<1)||(k>m));

**delcow(matr,n,m,k-1);**

 printf(" Matrica Result after delcow %4d\n",k);

**printmatr(matr,n,m);**

**delrow(matr,n,m,l-1);**

 printf(" Matrica Result after delrow %4d\n",l);

**printmatr(matr,n,m);**

 getch();

return 0;

}

**Пример работы программы с проверкой вводимых размеров**



#  Задания для решения на семинаре

1. Сформировать одномерный массив целых чисел размером не более 100 элементов. В сформированном массиве определить сумму четных элементов массива, произведение его отрицательных чисел и среднее арифметическое его элементов, кратных трем. Использовать подпрограммы.

Анализ показал, что для решения задачи нужно:

* Сформировать массив.
* Вывести исходный массив.
* Найти сумму четных элементов.
* Произведение отрицательных элементов.
* Среднее арифметическое элементов, кратных трем.

Для решения всех подзадач можно использовать поэлементную обработку массива. Кроме того, все подзадачи реализуем с помощью подпрограмм.

Для формирования разработаем подпрограмму-процедуру. Используем датчик случайных чисел, куда будем передавать имя массива, его размер и диапазон формируемых значений для формулы

El= rand()%(b-a+1)+a;

Для печати массива тоже используем подпрограмму процедуру. Так как массив может быть большим, для компактной печати будем делить его при выводе на последовательности по 10 элементов.

Все остальные подпрограммы считают численные значения, поэтому можно использовать подпрограммы-функции.

**Структурная схема программы**

**Текст программы**

**#include <stdio.h>**

**#include <conio.h>**

**#include <stdlib.h>**

**#include <time.h>**

**void printmas(int massiv[],int n)**

**{ int i;**

**for(i=0;i<n;i++)**

 **if((i+1)%10==0)**

 **printf("%5d\n",massiv[i]);**

 **else printf("%5d",massiv[i]);**

**printf("\n");**

**}**

**void formmas(int massiv[],int n,int a,int b)**

**{int i;**

 **for(i=0;i<n;i++)**

 **massiv[i]=rand()%(b-a+1)+a;**

**}**

**int sumchet(int massiv[],int n,int &k)**

**{int i,sum=0;**

**k=0;**

 **puts("chetnue elementy");// отладочная печать**

 **for(i=0;i<n;i++)**

 **if((massiv[i]%2==0)&&(massiv[i]!=0))**

 **{**

 **sum=sum+massiv[i];k++;**

 **printf("%5d",massiv[i]);**

 **}**

 **printf("\n");**

 **return sum;**

**}**

**int proizotr(int massiv[],int n,int &k)**

**{int i,p=1;k=0;**

 **puts("Negativ elements");// отладочная печать**

 **for(i=0;i<n;i++)**

 **if(massiv[i]<0)**

 **{**

 **p=p\*massiv[i];k++;**

 **printf("%5d",massiv[i]);**

**}**

**printf("\n");**

 **return p;**

**}**

**float sredkr3(int massiv[],int n,int &k)**

**{int i;**

 **float sum=0;k=0;**

 **puts("Elements kratnye 3");// отладочная печать**

 **for(i=0;i<n;i++)**

 **if((massiv[i]%3==0)&&(massiv[i]!=0))**

 **{**

 **k++;**

 **sum=sum+massiv[i];**

 **printf("%5d",massiv[i]);**

 **}**

 **printf("\n");**

 **if(k!=0)sum=sum/k;**

 **return sum;**

**}**

**int main()**

**{int mas[100],i,j,n,k;**

**srand((unsigned)time(NULL));**

**puts("Input n<=100");**

**scanf("%d",&n);**

**formmas(mas,n,-25,30);**

**puts(" Inputed Massiv");**

**printmas(mas,n);**

**int chet=sumchet(mas,n,k);**

 **if (k!=0)printf("Summa chetnyx elementov=%6d\n",chet);**

 **else puts("chetnyx elementov v massive net");**

**int p=proizotr(mas,n,k);**

**if(k==0)puts("otricatelnyx elementov v massive net");**

**else printf("Proizvedenie otricatelnyx elementov=%6d\n",p);**

**float sr=sredkr3(mas,n,k);**

**if(k!=0)**

 **printf("Srednee arifmeticheskoe elementov kratnyx 3=%6.3f\n",sr);**

**else puts("KRATNYX 3 elementov v massive net");**

**getch();**

**return 0;**

**}**

****  **Пример работы программы**

# Сформировать матрицу D[N][N], где n<=20 и вводится с клавиатуры. В исходной матрице поменять местами строку и столбец, на пересечении которых находится максимальный элемент матрицы. Если их несколько – берется первый из них. Вывести на экран исходную матрицу, найденный максимальный элемент и полученную матрицу. Использовать подпрограммы.

Анализ показал, что задачу можно декомпозировать на следующие подзадачи:

* Формирование матрицы.
* Вывод исходной матрицы.
* Нахождение максимального элемента и его координат.
* Обмен строки и столбца.
* Вывод полученной матрицы.

Для реализации всех подзадач можно использовать подпрограммы процедуры. Этот выбор обоснован тем, что все подзадачи выполняют с матрицей определенные манипуляции. А подзадача нахождения максимума должна найти и вернуть в основную программу не только найденный максимум, но и его координаты в матрице. Поэтому, три значения можно вернуть только через список параметров, используя параметры ссылки.

**Структурная схема**

При разработке алгоритма перестановки следует обратить внимание на то, что без вспомогательных массивов и в одном цикле перестановка возможна только, если максимум находится на главной или побочной диагонали. В остальных случаях перестановка выполняется некорректно и существенно зависит от места расположения максимального элемента.

В представленном ниже варианте реализации перестановка выполняется с применением вспомогательных массивов для строки и столбца, участвующих в обмене.

**Текст программы**

#include<stdio.h>

#include<stdlib.h>

#include<time.h>

**void printm(int matr[][20],int l,char \*s )**

{int i,g;

 puts(s);

for (i=0;i<l;i++)

for (g=0;g<l;g++)

 printf ("%3d%c",matr[i][g],(g==(l-1))?'\n':' ');

}

**void formrand(int matr[][20],int n)**

{ int i,j;

 srand((unsigned)time(NULL) );

for (i=0;i<n;i++)

for (j=0;j<n;j++)

 matr[i][j]=rand()%50-rand()%20;

}

void formkey(int matr[][20],int n)

{ int i,j;

 printf("Input %5d strok po %5d numbers\n",n,n);

for (i=0;i<n;i++)

for (j=0;j<n;j++)

 scanf("%d",&matr[i][j]);

}

void Maxij(int matr[][20],int n,int&max,int&imax,int&jmax)

{ int i,j;

 max=matr[0][0];

imax=jmax=0;

for (i=0;i<n;i++)

for (j=0;j<n;j++)

if (matr[i][j]>max)

 {

 max=matr[i][j];

 imax=i;

 jmax=j;

 }

}

void pereform1(int matr[][20],int n,int imax,int jmax)

{ int i,j,k;

 int a[20];int b[20];// вспомогательные массивы

 for (i=0;i<n;i++)// цикл формированиявспомогательных массивов

 {

 a[i]=matr[i][jmax];

 b[i]=matr[imax][i];

 }

 puts("Masiv a stolbec");

 for(i=0;i<n;i++)

 {

 printf("%4d",a[i]);

 }

 puts("\nMasiv b string");

 for(i=0;i<n;i++)

 {

 printf("%4d",b[i]);

 }

 printf("\n");

// собственно, обмен

 for (i=0;i<n;i++)

 {

 matr[imax][i]=a[i];

 }

 printf("\n");

 for(i=0;i<n;i++)

 {

 matr[i][jmax]=b[i];

 }

 printf("\n");

}

int main()

{

int a[20][20],i,n,m,c,im,jm,max;

 puts("input n<=20");

 scanf("%d",&n);

 puts("Input 1 to formrand");

 puts("Input 2 to formkey");

 scanf("%d",&c);

if (c==1) formrand(a,n);

if (c==2) formkey(a,n);

 printm(a,n,"Inputed matrix ");

 Maxij(a,n,max,im,jm);

 printf("\n\nMax=%5d,i=%4d, j=%4d\n\n",max,im+1,jm+1);

 pereform1(a,n,im,jm);

 printm(a,n," Formated matrix after change ");

return 0;

}

**Примеры работы программы**

****



# Задания для тренировки .

1. В заданной матрице N\*M определить сумму элементов строк, номера которых записаны в одномерном массиве L(K) (K<N) целочисленных элементов в произвольном порядке. Номера вводятся с клавиатуры и не повторяются. Вывести на экран исходную и полученную матрицы, а также массив L. Предусмотреть контроль ввода всех размеров и номеров строк. Использовать подпрограммы.
2. Из заданной целочисленной матрицы N\*M (M>N) удалить столбцы, в которых встречается более трех четных элементов. Вывести на экран исходную, полученную матрицы и найденное количество четных элементов каждого столбца. Использовать подпрограммы.
3. В вещественной матрице размером n x m определить наименьший среди элементов заштрихованной области (см. рис.), считая его единственным. Вывести найденный элемент, его координаты и сумму индексов найденного элемента. При вводе чисел p и q проверить истинность неравенства: 1 ≤ p < q ≤ m. Использовать подпрограммы.
4. В каждой из заштрихованных частей вещественной квадратной матрицы порядка n определить количество элементов, меньших заданного числа q (см. рис., 1 < k < n, k ∈ **N**). Если в нижней части матрицы таких элементов больше, чем в верхней, то заменить элементы главной диагонали на 0. Вывести на экран матрицу в исходном и полученном виде, а также найденные элементы и количественные значения. Использовать подпрограммы.