

Объединения

Хранение разнотипных данных в одной области памяти.

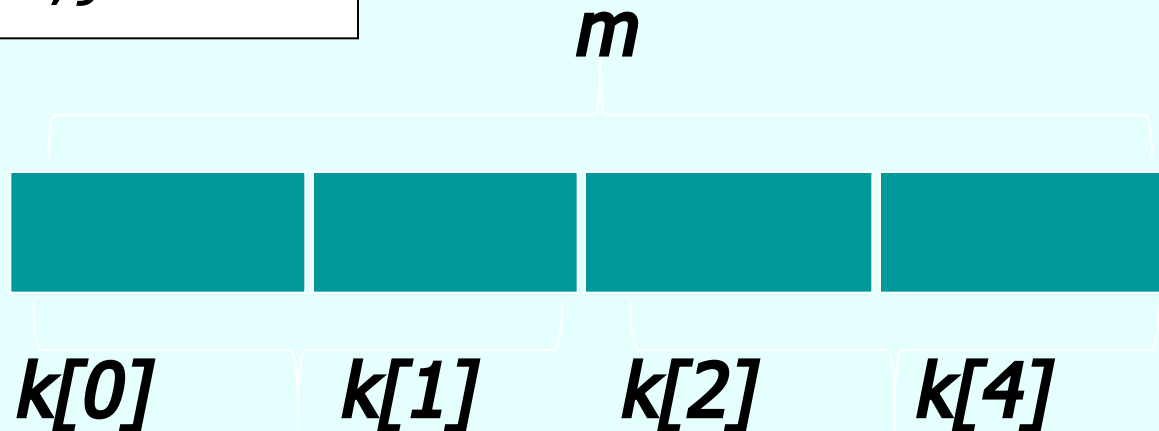
Синтаксис:

```
union [имя] {  
    тип поле1;  
    тип поле2;  
    ...  
}
```

Размер объединения - это размер его максимального элемента.

В каждый момент времени может быть сохранен только один из элементов объединения.

```
union MyUnion{  
char k[4];  
unsigned int m;}
```



Перечисления

Тип для работы с целыми константами,

Синтаксис:

```
enum [ имя] { Имя1, Имя2, ... };
```

```
enum number { one, two, three };
```

```
enum number { one, two, three };
```

Элементы перечисления

- Определено перечисление с именем *number*
- Определены три целых константы *one, two, three*
- Им присвоены значения по умолчанию
const one = 0;
const two = 1;
const three = 2;

```
number k = one;  
int i =two;  
k = i;  
k = number(i);  
i = k;  
k = 4;  
...
```

```
number key;  
...  
switch(key){  
case one: ...; break;  
case two: ...; break;  
case three: ...;  
}
```

Значения элементов перечисления можно задавать и явно:

```
enum number { one=1,  
              two=2,  
              three=3 };
```

Задаваемые значения **необязательно** должны быть

- различными,
- положительными,
- идти в возрастающем порядке.

Функции

Синтаксис

Описание

```
[тип возвращаемого значения] Имя_Функции  
    ( [тип Аргумент1,  
      тип Аргумент2, ...] )  
{  
    операторы;  
    [return Возвращаемое_значение];  
}
```

Описание функции может быть выполнено до функции *main* :

```
int Max3(int x, int y, int z)  
{ ... }  
int main(...) {  
... }
```

либо после функции *main* :

```
int Max3(int , int , int );  
int main(...) {  
... }  
int Max3(int x, int y, int z)  
{ ... }
```


Вызов функции

```
int Max3(int x, int y, int z)  
{ ... }  
int main(...) {  
int k,l;  
  
...  
int f=5; z=4; m=15;  
k = Max3(f,z,m);  
  
...  
l=Max3(4,11,3);  
}
```

**Формальные
аргументы
(параметры)**

**Замена
формальных
аргументов**

```
int Max3(int x, int y, int z) {
```

```
    int max = x;
```

```
    if (max < y) max = y;
```

```
    if (max < z) max = z;
```

```
    return max; }
```

```
int main() {
```

```
    int k, l, f = 5, z = 4, m = 15;
```

```
    k = Max3(f, z, m);
```

```
    l = Max3(4, 11, 3);
```

```
    printf ("\nk=%4d, l=%4d", k,
```

```
    }
```

x=f

y=z

z=m

x=4

y=11

z=3

Функции и программный стек

```
int Max3(int x, int y, int z) {  
    int max = x;  
    if (max < y) max = y;  
    if (max < z) max = z;  
    return max; }  
  
int main(...) {  
    int k, l, f = 5, z = 4, m = 15;  
    k = Max3(f, z, m);  
    l = Max3(4, 11, 3);  
    printf ("\nk = %4d, l = %4d", k, l);  
}
```

Глобальные и локальные переменные

Переменные, описанные внутри блока программы, ограниченного открывающейся и закрывающейся фигурными скобками называются **локальными** переменными

```
float MyFunc(int x)
{ float z = x;
  return z*z;
}
```

```
int z = 1;
int MyFunc(int x){
int k=7;
if (x>0) x++;
z++;
return x;
}
int main()
```

```
{
int k = 4;
z++;
int x = 8;
k=MyFunc(k);
x++;
printf("%d %d %d...\n",x,z,k);
return 0;
}
```

z=3

k=5

x=9

x=5

k=7

9 3 5...

Параметры функции

- Параметры функции перечисляются в круглых скобках после имени функции:

```
int function1 (int k, int f)
```

```
float function2 (float z)
```

```
int function3 (char m)
```

```
void function (char *s)
```

- Функция может не иметь параметров:

```
int function4 ()
```

- При вызове функции формальные параметры заменяются указанными значениями (фактическими параметрами):

```
int m = 0, m1 = 1; ...
```

```
int float z = sin(M_PI); ...
```

```
float ...
```

```
int char d[20] = "Пример строки";
```

```
function
```

```
int f = function4 ();
```

- Передача одномерного массива параметром:

```
// Функция печати массива  
void Show( int *x, int n, char* t) {  
    printf("\n%s\n",t);  
    for(int i=0;i<n;i++)  
        printf("%4d",x[i]);  
}
```



```
// Функция создания массива  
int *Create( int *x, int n){  
    x = (int*)malloc(sizeof(int)*n);  
    for(int i=0;i<n;i++)  
        x[i] = rand()%20-rand()%20;  
    return x;  
}
```

```
// Функция поиска количества элементов  
// массива равных 0  
int Zero (int *x, int n)  
{ int z = 0;  
    for(int i=0;i<n;i++)  
        if(x[i]==0) z++;  
    return z; }
```

```
// Функция изменения массива – замена всех
// положительных элементов их индексами
int * Change( int *x, int n) {
    for(int i=0;i<n;i++)
        if (x[i]>0) x[i]=i;
    return x;
}

int main () {
    srand(time(NULL));
    int *mas, *mas1;
    mas = Create(mas,20);
    mas1 = Create(mas1,25);
```

```
// Вызов функции печати
Show( mas,20," Первый массив:" );
// Вызов функции подсчета нулей
printf("\nКоличество нулей - %d", Zero(mas,20) );
// Вызов функции печати
Show( mas1,25," Второй массив:" );
// Вызов функции подсчета нулей
printf("\n Количество нулей - %d",Zero(mas1,25) );
// Вызов функции изменения массива
mas = Change( mas,20 );
// Вызов функции изменения массива
mas1 = Change( mas1,25 );
```

```
// Вызов функции печати
```

```
Show(mas,20,"Первый массив после изменения:");
```

```
// Вызов функции печати
```

```
Show(mas1,25,"Второй массив после изменения" );
```

```
free(mas);
```

```
free(mas1);
```

```
return 0;
```

```
}
```

Передача матрицы параметром

```
// Вычисление суммы элементов строки с  
// номером num матрицы x  
int Sum( int **x, int m, int num ) {  
    int sum = 0;  
    for ( int i=0;i<m;i++)  
        sum+=x[num][i];  
    return sum;  
}
```

```
int main() {  
int n,m;  
printf(" Вводите количество строк: ");  
scanf("%d",&n);  
printf(" Вводите количество столбцов: ");  
scanf("%d",&m);  
int **matr = new int*[n];  
for (int i=0;i<n;i++)  
    matr[i]=new int[m];  
for(i=0;i<n;i++) {  
    for(int j=0;j<m;j++)
```

```
{ matr[i][j] = rand()%20;
  printf("%3d",matr[i][j]); }
printf("\n");
}
printf(" Вводите номера строк:");
int k,l;
scanf("%d%d",&k,&l);
if (k>0&&k<n&&l>0&&k<n) {
// Вызовы функции Sum
printf("Сумма в %d строке %d\n",k,Sum(matr,m,k));
printf(«Сумма в %d строке %d\n",l,Sum(matr,m,l));
}
```

```
else printf(" Ошибка ввода данных");  
for(i=n-1;i>=0;i--)  
    delete [] matr[i];  
delete [] matr;  
}
```


Возвращаемое значение

- Тип возвращаемого значения указывается перед именем функции

[тип] имя функции (...)

- Возвращаемое значение передается в основную программу оператором *return:*

return [возвращаемое значение]

- Оператор *return* *всегда* заканчивает выполнение функции

```
void func1(int x) {  
    int k = 8;  
    return;  
    x++; }  
}
```

**Невыполнимый
код**

- Если функция не содержит оператора *return*, ее выполнение заканчивается при достижении **закрывающей фигурной скобки**:

```
void func2(int *x, int n) {  
    for(int i=0;i<n;  
        printf("%3d",  
    }  
}
```

**Конец
выполнения
функции**

- Тип возвращаемого значения должен совпадать с типом значения, возвращаемого *return*:

```
int func1(float x, float y) {  
  if (x==y) return 0;  
  else if (x<y) return -1;  
    else return 1.;  
}
```

Ошибка!!!

- По умолчанию тип возвращаемого значения – *int*:

```
func1(float x, float y) {  
  if (x==y) return 0;  
  else if (x<y) return -1;  
    else return 1;  
}
```

- Функция всегда возвращает не более одного значения.

Параметры по ссылке

Если возникла ситуация, при которой необходимо вернуть из функции **более, чем одно** значение используются ***параметры по ссылке.***

При передаче в функцию **адреса переменной** изменения значений такой переменной производится непосредственно **в программном стеке.**

```

#include <conio.h>
#include <stdio.h>
void swap(int* x, int* y)
{ int temp = *x;
  *x=*y;
  *y=temp; }
void main() {
int n=7,m=8;
swap( &n,&m);
printf("\%d \%d", n,m);
}

```

Область описания

переменных

~~x=&y~~ ~~y=&m~~
n m
temp=7

Стек

n=8 m=7

Дан массив из n элементов. Удалить из массива все отрицательные элементы.

```
#include <conio.h>  
#include <stdio.h>  
#include <stdlib.h>  
void show( int *x, int n) {  
for (int i=0;i<n;i++)  
    printf("%4d",x[i]);  
printf("\n");  
}  
int *Delete( int *x, int*n) {
```

```
int kol = 0;
for (int i=0;i<*n; i++)
    if (x[i]<0) kol++;
int *x1 = new int[*n-kol];
int j;
for (i=0,j=0;i<*n;i++)
    if (x[i]>=0) {x1[j]=x[i]; j++;}
delete [] x;
*n-=kol;
return x1;
}
int main() {
clrscr();
```



```
int n;  
printf(" Введите количество элементов: ");  
scanf("%d",&n);  
int *mas = new int[n];  
for (int i=0;i<n;i++)  
    mas[i] = rand()%20-rand()%20;  
show(mas,n);  
mas = Delete(mas,&n);  
show(mas,n);  
  
}
```

Параметры по умолчанию

В функциях Си разрешено использовать **параметры по умолчанию**.

Написать функцию, которая считает количество отрицательных (или положительных) элементов матрицы.

```
int PN( int** x, int n, int m, int flag = 0) {  
    int ch = 0;  
    for(int i=0;i<n;i++)  
        for(int j=0;j<m;j++)
```

```
if (x[i][j]>0)
    { if (!flag) ch++;}
    else if (flag && x[i][j]) ch++;
return ch;
}
```

```
int main(...) {
int n,m;
printf(" Введите количество строк: ");
scanf("%d",&n);
int **matr = new int*[n];
```

```
printf(" Введите количество столбцов: ");
scanf("%d",&m);
for (int i=0;i<n;i++)
    matr[i] = new int[m];
for(i=0;i<n;i++) {
    for(int j=0;j<m;j++)
        { matr[i][j] = rand()%20-rand()%20;
          printf("%4d",matr[i][j]);
        }
    printf("\n");
}
```

```
printf(" Отрицательных - %d\n", PN(matr,n,m,1));  
printf(" Положительных - %d\n", PN(matr,n,m));  
}
```

Указатель на функцию

Как на любой объект программы можно объявить **указатель на функцию**

Синтаксис:

[тип возвращаемого значения] (* имя)
(перечисление типов параметров
функции)

Написать функцию Си, выводящую на экран таблицу значений **заданной** математической функции на **заданном** интервале с **заданным** шагом.

```
#include <conio.h>
#include <stdio.h>
#include <math.h>
float f1( float x)
    {return sin(x)+x*x; }
float f2( float x)
    { return 2*x*x-3*x-10;}

```

```

float f3( float x)
{ return 1/x+3*x; }
void table( float (*f)(float), float a, float b, int i) {
float step = (b-a)/(i-1);
printf("*****\n");
printf("* x          * f(x)          *\n");
printf("*****\n");
float x = a;
for(int k=0;k<i;k++)
{ printf("* %-16.4f* %-18.4f*\n",x,f(x));
x+=step; }
printf("*****\n");
}

```



```
int main(...) {  
float (*f)(float);  
f=&f1;  
printf(" Первая функция:\n");  
table( f,0,M_PI,10);  
f = &f2;  
printf(" Вторая функция:\n");  
table( f,0,20,18);  
f = &f3;  
printf(" Третья функция:\n");  
table( f,1,5,20);  
}
```