Масив.

Подредена, последователна и **непрекъсната** в паметта редица от еднакви по тип елементи.
Например, поредица от цели числа: 5 21 72 8 1 5 5 -20

Декларация на масив.

```
// Заделя памет за масив с определен размер:
<тип> <име>[<размер>];
// Забележка: Размера на масива трябва да е константа, известна по
// време на компилация.
// Забележка 2: Веднъж създаден масива, той не може да бъде
// преоразмеряван. (Не можем да му променим размера)
```

Дефиниция на масив.

```
// Създава масив с елементи подадените в скоби цели числа
// Размера на масива е броят подадени елементи в скобите
int arr[] = { 4, 7, 23, 18 };

// Създава масив с размер 6, на който първите три елемента са 4, 2 и 6
// останалите елементи са нули. (Масива изглежда така: [4, 2, 6, 0, 0, 0])
int arr[6] = { 4, 2, 6, };
```

Достъпване на елемент в масив.

Индекс - позиция в масив, започваща от 0.

```
cout << arr[0];          // <име>[<индекс>]
```

Промяна на стойността на елемент в масив.

```
arr[2] = 5;              // <име>[<индекс>] = стойност;
```

Какво всъщност представляват масивите.

Всъщност **arr** е *адреса,* в паметта, на първия елемент на масива.
arr[i] казва *“Отиди на адреса **arr** и се отмести **i** пъти напред”*.

Подаване на масив като аргумент на функция.

```
void printArray(const int arr[], int size);
```

Тъй като **arr** е адрес в паметта, то когато използваме arr във функцията ще работим **ДИРЕКТНО** с масива и **НЯМА** да се създаде копие на масива.

```
// Забележка: При подаването на масива като аргумент, не е нужно да
// уточняваме размера му в квадратни скоби, тъй като ние му подаваме
// адреса на първия елемент.
// Забележка 2: Тъй като подаваме само адреса на първия елемент,
// трябва да знаем кога сме достигнали края на масива. Това може да
// се постигне като подадем броя на елементите като аргумент или ако
// фиксираме някой специален елемент да е последен.
```

Още съвети...

- Достъпването на елементи извън границите на масива води до нежелани ефекти.
  Не правете това: `arr[-3]` *или* `arr[n]` *където n е размера на масива.*

- Ако функцията, на която подаваме масива, **не** се очаква да променя масива, то подаваме масива като **константа**.
  Например: `void printArray(const int arr[], int size);`

- Итерирането през елементите на масив става чрез цикъл.

```
for (int i = 0;  i < arrSize;  i++)
        cout << arr[i];      // Отпечатва всеки елемент на масива
```