



**СГУГиТ**

СИБИРСКИЙ  
ГОСУДАРСТВЕННЫЙ  
УНИВЕРСИТЕТ  
ГЕОСИСТЕМ И ТЕХНОЛОГИЙ

# Массив



# СГУГиТ

СИБИРСКИЙ  
ГОСУДАРСТВЕННЫЙ  
УНИВЕРСИТЕТ  
ГЕОСИСТЕМ И ТЕХНОЛОГИЙ

**Массив — это область памяти, где могут последовательно храниться несколько значений.**

Возьмем группу студентов из десяти человек. У каждого из них есть фамилия. Создавать отдельную переменную для каждого студента — не рационально. Создадим массив, в котором будут храниться фамилии всех студентов.



# СГУГиТ

СИБИРСКИЙ  
ГОСУДАРСТВЕННЫЙ  
УНИВЕРСИТЕТ  
ГЕОСИСТЕМ И ТЕХНОЛОГИЙ

## Пример инициализации массива

```
string students[10] = {  
"Иванов", "Петров", "Сидоров",  
"Ахмедов", "Ерошкин", "Выхин",  
"Андеев", "Вин Дизель", "Картошкин",  
"Чубайс"  
};
```



# СГУГиТ

СИБИРСКИЙ  
ГОСУДАРСТВЕННЫЙ  
УНИВЕРСИТЕТ  
ГЕОСИСТЕМ И ТЕХНОЛОГИЙ

## Описание синтаксиса

Массив создается почти так же, как и обычная переменная. Для хранения десяти фамилий нам нужен массив, состоящий из 10 элементов. Количество элементов массива задается при его объявлении и заключается в квадратные скобки.



# СГУГиТ

СИБИРСКИЙ  
ГОСУДАРСТВЕННЫЙ  
УНИВЕРСИТЕТ  
ГЕОСИСТЕМ И ТЕХНОЛОГИЙ

Чтобы описать элементы массива сразу при его создании, можно использовать фигурные скобки. В фигурных скобках значения элементов массива перечисляются через запятую. В конце закрывающей фигурной скобки ставится точка с запятой.

Попробуем вывести наш массив на экран с помощью оператора **cout**.



# СГУГиТ

СИБИРСКИЙ  
ГОСУДАРСТВЕННЫЙ  
УНИВЕРСИТЕТ  
ГЕОСИСТЕМ И ТЕХНОЛОГИЙ

```
#include <iostream>
```

```
#include <string>
```

```
int main()
```

```
{
```

```
std::string students[10] = {
```

```
"Иванов", "Петров", "Сидоров",
```

```
"Ахмедов", "Ерошкин", "Выхин",
```

```
"Андреев", "Вин Дизель", "Картошкин", "Чубайс"
```

```
};
```

```
std::cout << students << std::endl; // Пытаемся вывести весь
```

```
массив непосредственно
```

```
return 0;
```

```
}
```



# СГУГиТ

СИБИРСКИЙ  
ГОСУДАРСТВЕННЫЙ  
УНИВЕРСИТЕТ  
ГЕОСИСТЕМ И ТЕХНОЛОГИЙ

Скомпилируйте этот код и посмотрите, на результат работы программы. Готово? А теперь запустите программу еще раз и сравните с предыдущим результатом.

В моей операционной системе вывод был следующим:

- Первый вывод: **0x7ffff8b85820**
- Второй вывод: **0x7fff7a335f90**
- Третий вывод: **0x7ffff847eb40**



# СГУГиТ

СИБИРСКИЙ  
ГОСУДАРСТВЕННЫЙ  
УНИВЕРСИТЕТ  
ГЕОСИСТЕМ И ТЕХНОЛОГИЙ

Мы видим, что выводится адрес этого массива в оперативной памяти, а никакие не «Иванов» и «Петров».

Дело в том, что при создании переменной, ей выделяется определенное место в памяти. Если мы объявляем переменную типа `int`, то на машинном уровне она описывается двумя параметрами — ее адресом и размером хранимых данных.





# СГУГиТ

СИБИРСКИЙ  
ГОСУДАРСТВЕННЫЙ  
УНИВЕРСИТЕТ  
ГЕОСИСТЕМ И ТЕХНОЛОГИЙ

Массивы в памяти хранятся таким же образом. Массив типа `int` из 10 элементов описывается с помощью адреса его первого элемента и количества байт, которое может вместить этот массив. Если для хранения одного целого числа выделяется 4 байта, то для массива из десяти целых чисел будет выделено 40 байт.



# СГУГиТ

СИБИРСКИЙ  
ГОСУДАРСТВЕННЫЙ  
УНИВЕРСИТЕТ  
ГЕОСИСТЕМ И ТЕХНОЛОГИЙ

Так почему же, при повторном запуске программы, адреса различаются? Это сделано для защиты от **атак переполнения буфера**. Такая технология называется **рандомизацией адресного пространства** и реализована в большинстве популярных ОС.

Попробуем вывести первый элемент массива — фамилию студента Иванова.



# СГУГиТ

СИБИРСКИЙ  
ГОСУДАРСТВЕННЫЙ  
УНИВЕРСИТЕТ  
ГЕОСИСТЕМ И ТЕХНОЛОГИЙ

```
#include <iostream>
#include <string>
int main()
{
    std::string students[10] = {
        "Иванов", "Петров", "Сидоров",
        "Ахмедов", "Ерошкин", "Выхин",
        "Андреев", "Вин Дизель", "Картошкин",
        "Чубайс"
    };
    std::cout << students[0] << std::endl;
    return 0;
}
```



# СГУГиТ

СИБИРСКИЙ  
ГОСУДАРСТВЕННЫЙ  
УНИВЕРСИТЕТ  
ГЕОСИСТЕМ И ТЕХНОЛОГИЙ

Смотрим, компилируем, запускаем. Убедились, что вывелся именно «Иванов». Заметьте, что нумерация элементов массива в C++ начинается с нуля. Следовательно, фамилия первого студента находится в `students[0]`, а фамилия последнего — в `students[9]`.

В большинстве языков программирования нумерация элементов массива также начинается с нуля.



# СГУГиТ

СИБИРСКИЙ  
ГОСУДАРСТВЕННЫЙ  
УНИВЕРСИТЕТ  
ГЕОСИСТЕМ И ТЕХНОЛОГИЙ

Попробуем вывести список всех студентов. Но сначала подумаем, а что если бы вместо группы из десяти студентов, была бы кафедра их ста, факультет из тысячи, или даже весь университет? Ну не будем же мы писать десятки тысяч строк с `cout`?

Конечно же нет! Мы возьмем на вооружение циклы, о которых был написан **предыдущий урок**.



# СГУГиТ

СИБИРСКИЙ  
ГОСУДАРСТВЕННЫЙ  
УНИВЕРСИТЕТ  
ГЕОСИСТЕМ И ТЕХНОЛОГИЙ

## Вывод элементов массива через цикл

```
#include <iostream>
#include <string>
int main()
{
    std::string students[10] = {
        "Иванов", "Петров", "Сидоров",
        "Ахмедов", "Ерошкин", "Выхин",
        "Андреев", "Вин Дизель", "Картошкин", "Чубайс"
    };
    for (int i = 0; i < 10; i++) {
        std::cout << students[i] << std::endl;
    }
    return 0;
}
```



# СГУГиТ

СИБИРСКИЙ  
ГОСУДАРСТВЕННЫЙ  
УНИВЕРСИТЕТ  
ГЕОСИСТЕМ И ТЕХНОЛОГИЙ

Если бы нам пришлось выводить массив из нескольких тысяч фамилий, то мы бы просто увеличили конечное значение счетчика цикла — строку `for (...; i < 10; ...)` заменили на `for (...; i < 10000; ...)`.

Заметьте что счетчик нашего цикла начинается с нуля, а заканчивается девяткой.



# СГУГиТ

СИБИРСКИЙ  
ГОСУДАРСТВЕННЫЙ  
УНИВЕРСИТЕТ  
ГЕОСИСТЕМ И ТЕХНОЛОГИЙ

Если вместо оператора строгого неравенства —  $i < 10$  использовать оператор «меньше, либо равно» —  $i \leq 10$ , то на последней итерации программа обратится к несуществующему элементу массива — `students[10]`.





# СГУГиТ

СИБИРСКИЙ  
ГОСУДАРСТВЕННЫЙ  
УНИВЕРСИТЕТ  
ГЕОСИСТЕМ И ТЕХНОЛОГИЙ

Это может привести к **ошибкам сегментации** и аварийному завершению программы. Будьте внимательны — подобные ошибки бывает сложно отловить.

Массив, как и любую переменную можно не заполнять значениями при объявлении.



СГУГиТ

СИБИРСКИЙ  
ГОСУДАРСТВЕННЫЙ  
УНИВЕРСИТЕТ  
ГЕОСИСТЕМ И ТЕХНОЛОГИЙ

## Объявление массива без инициализации

```
string students[10];
```

```
// или
```

```
string teachers[5];
```



# СГУГиТ

СИБИРСКИЙ  
ГОСУДАРСТВЕННЫЙ  
УНИВЕРСИТЕТ  
ГЕОСИСТЕМ И ТЕХНОЛОГИЙ

Элементы такого массива обычно содержат в себе «мусор» из выделенной, но еще не инициализированной, памяти. Некоторые компиляторы, такие как GCC, заполняют все элементы массива нулями при его создании.



# СГУГиТ

СИБИРСКИЙ  
ГОСУДАРСТВЕННЫЙ  
УНИВЕРСИТЕТ  
ГЕОСИСТЕМ И ТЕХНОЛОГИЙ

При создании статического массива, для указания его размера может использоваться только константа. Размер выделяемой памяти определяется на этапе компиляции и не может изменяться в процессе выполнения.



# СГУГиТ

СИБИРСКИЙ  
ГОСУДАРСТВЕННЫЙ  
УНИВЕРСИТЕТ  
ГЕОСИСТЕМ И ТЕХНОЛОГИЙ

```
int n;
```

```
cin >> n;
```

```
string students[n]; /* Неверно */
```

Выделение памяти в процессе выполнения возможно при работе с **динамическими массивами**. Но о них немного позже.

Заполним с клавиатуры пустой массив из 10 элементов.



# СГУГиТ

СИБИРСКИЙ  
ГОСУДАРСТВЕННЫЙ  
УНИВЕРСИТЕТ  
ГЕОСИСТЕМ И ТЕХНОЛОГИЙ

## Заполнение массива с клавиатуры

```
#include <iostream>
#include <string>
using std::cout;
using std::cin;
using std::endl;
int main()
{
    int arr[10]; // Заполняем массив с клавиатуры
    for (int i = 0; i < 10; i++) {
        cout << "[" << i + 1 << "]" << ": ";
    }
}
```



# СГУГиТ

СИБИРСКИЙ  
ГОСУДАРСТВЕННЫЙ  
УНИВЕРСИТЕТ  
ГЕОСИСТЕМ И ТЕХНОЛОГИЙ

```
cin >> arr[i];  
}  
// И выводим заполненный массив.  
cout << "\nВаш массив: ";  
for (int i = 0; i < 10; ++i) {  
cout << arr[i] << " ";  
}  
cout << endl;  
return 0;  
}
```



# СГУГиТ

СИБИРСКИЙ  
ГОСУДАРСТВЕННЫЙ  
УНИВЕРСИТЕТ  
ГЕОСИСТЕМ И ТЕХНОЛОГИЙ

```
Терминал — zsh — 80x24
ssh
zsh
selevit ~ % ./foo
[1]: 4
[2]: 5
[3]: 3
[4]: 3
[5]: 5
[6]: 2
[7]: 8
[8]: 4
[9]: 4
[10]: 9
Ваш массив: 4 5 3 3 5 2 8 4 4 9
selevit ~ %
```





# СГУГиТ

СИБИРСКИЙ  
ГОСУДАРСТВЕННЫЙ  
УНИВЕРСИТЕТ  
ГЕОСИСТЕМ И ТЕХНОЛОГИЙ

Если у вас возникают проблемы при компиляции исходников из уроков — внимательно прочитайте ошибку компилятора, попробуйте проанализировать и исправить ее. Если вы нашли ошибку в коде — **напишите об этом в комментариях к уроку.**