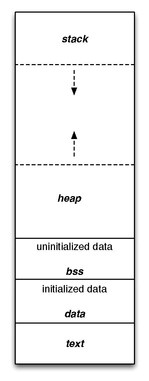
Static

**Static -** это ключевое слово в C++, используемое для придания элементу особых характеристик. Для статических элементов выделение памяти происходит только один раз и существуют эти элементы до завершения программы. Хранятся все эти элементы не в heap и не на stack, а в специальных сегментах памяти, которые называются *.data* и *.bss* (зависит от того инициализированы статические данные или нет). На картинке ниже показан типичный макет программной памяти.



Макет (типичный)

Ниже приведена схема, как и где используется **static** в программе.

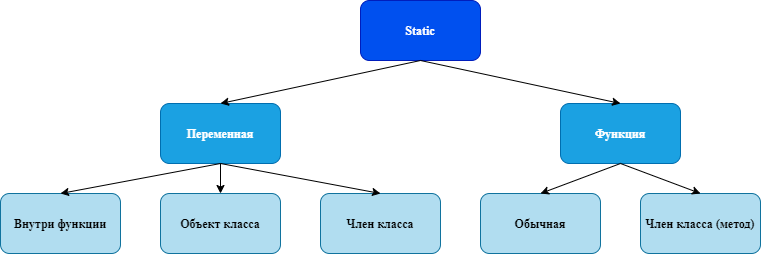


Схема (нетипичная)

Статические переменные внутри функции

Статические переменные при использовании внутри функции инициализируются только один раз, а затем они сохраняют свое значение. Эти статические переменные хранятся в статической области памяти (*.data или .bss*), а не в стеке, что позволяет хранить и использовать значение переменной на протяжении всей жизни программы.

#**include** <iostream>

**void** **counter**() {

**static** **int** count = 0; // строка 4

std::cout << count++;

}

**int** **main**() {

**for** (**int** i = 0; i < 10; ++i) {

counter();

}

**return** 0;

}

Программа выведет: «0123456789»

#**include** <iostream>

**void** **counter**() {

**int** count = 0; // строка 4

std::cout << count++;

}

**int** **main**() {

**for** (**int** i = 0; i < 10; ++i) {

counter();

}

**return** 0;

}

Программа выведет: «0000000000»

Если не использовать **static**в *строке 4*, выделение памяти и инициализация переменной count происходит при каждом вызове функции *counter()*, и уничтожается каждый раз, когда функция завершается. **Но используя static мы инициализируем переменную один раз и потом сохраняем её на время работы всей программы!** Вот так-то.

### Статические функции

В этом случае вы говорите компилятору, что доступ к статическим функциям ограничен файлом, в котором они объявлены. И он имеет доступ только к функции *sum()* из *math.cpp* файла. Таким образом, используя **static** для функции, мы можем ограничить область видимости этой функции, и данная функция не будет видна в других файлах, если, конечно, это не заголовочный файл (.h).

Пример:

// extend\_math.cpp

**int** **sum**(**int** a, **int** b) { //функция sum

**int** some\_coefficient = 1;

**return** a + b + some\_coefficient;

}

// math.cpp

**int** **sum**(**int** a, **int** b) { //и она же опять?

**return** a + b;

}

// main.cpp

**int** **sum**(**int**, **int**); // declaration

**int** **main**() {

**int** result = sum(1, 2); //не приходит на ум, а что вызывать?

**return** 0;

}

//Не работат!

Для того чтобы исправить данную проблему, одну из функций мы объявим статической. Например эту:  
  
// extend\_math.cpp

**static** **int** **sum**(**int** a, **int** b) {

**int** some\_coefficient = 1;

**return** a + b + some\_coefficient;

}

В этом случае вы говорите компилятору, что доступ к статическим функциям ограничен файлом, в котором они объявлены. И он имеет доступ только к функции *sum()* из *math.cpp* файла. С заголовочными файлами это не работает.

Источники:

<https://habr.com/ru/post/527044/>