Оглавление

[Структура простой программы. 2](#_Toc157285217)

[Вывод текста. 2](#_Toc157285218)

[Типы данных в языке c++ 2](#_Toc157285219)

[Переменные. 2](#_Toc157285220)

[Эскейп последовательности. 3](#_Toc157285221)

[Ввод данных в приложение. 3](#_Toc157285222)

[Математические операции. 3](#_Toc157285223)

[Циклы. 4](#_Toc157285224)

[Break и Continue. 4](#_Toc157285225)

[Массивы теория. 4](#_Toc157285226)

[Инициализация массивов. 4](#_Toc157285227)

[Ключевое слово sizeof для работы с массивами. 4](#_Toc157285228)

[Генерация случайных чисел. 5](#_Toc157285229)

[Функции. 5](#_Toc157285230)

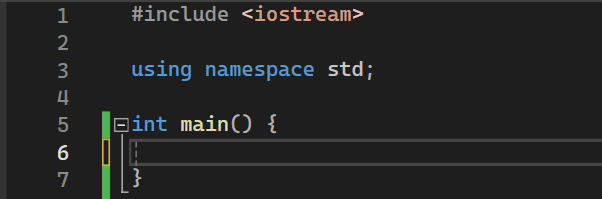
[Передача параметров в функцию. 6](#_Toc157285231)

[Прототипы функции. 6](#_Toc157285232)

[Ключевое слово Inline. 7](#_Toc157285233)

[Шаблонные функции. 7](#_Toc157285234)

Структура простой программы.



Вот так выглядит простая программа, первым делом мы подключаем библиотеку iostream.

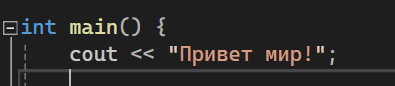
Дальше из библиотеки мы можем подключить неймспейс std для работы с консолью и прочим, как system в c#

После чего следует функция мейн, что и является точкой входа в программу.

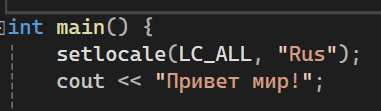
Вывод текста.

Для вывода текста в консоль используется команда cout << «Текст который хотите вывести»;

cout – скорее всего сокращение от c – C++ и out - вывод



Тут будет проблема, с++ не имеет в себе по стандарту русского языка и для его добавления стоит прописать следующее.



Через setlocale мы можем установить нужный нам язык.

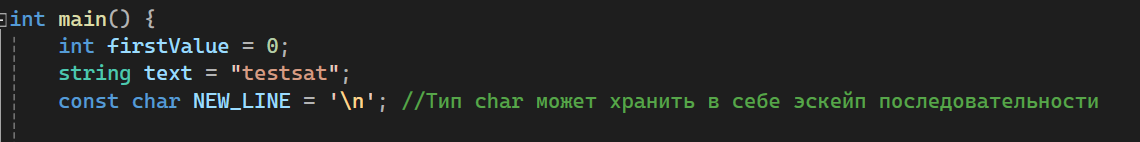
Типы данных в языке c++

По большей части, типы данных повторяют типы из c#, поэтому можно смотреть там.

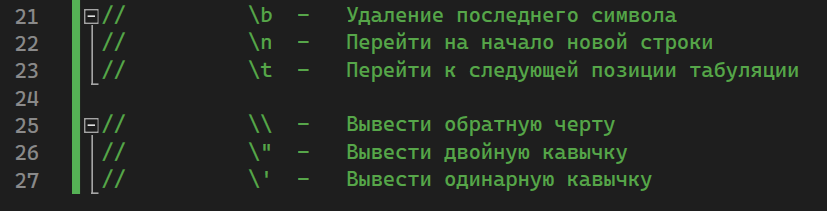
Переменные.

Объявление переменных идентично объявлению их на языке с# поэтому можно смотреть там.

Пример простых переменных.

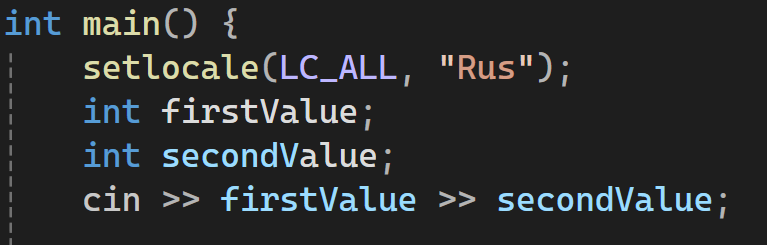


Эскейп последовательности.



Ввод данных в приложение.

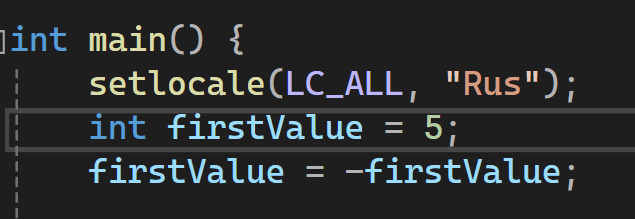
Для ввода данных используется команда cin в расшифровке c – c++, in – input



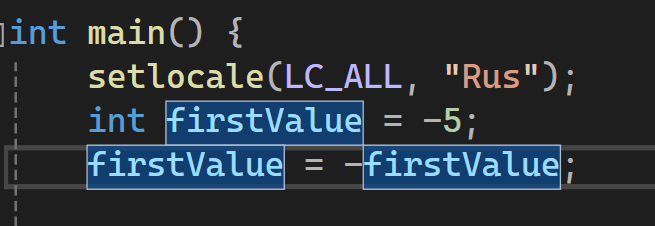
К примеру, тут мы сразу можем присвоить значение нескольким переменным, так-же мы можем присвоить значение одной переменной и всё это сразу делается в инт. С другими типами данных тоже работает.

Математические операции.

Одна из новых операций, это смена знака на противоположный:



Изначально знак был положительный, а после наших действий наша 5 превратилась в -5



А тут наша -5 превратиться в 5.

Циклы.

Идентичны циклам на c#, единственное отличие в цикле Foreach.

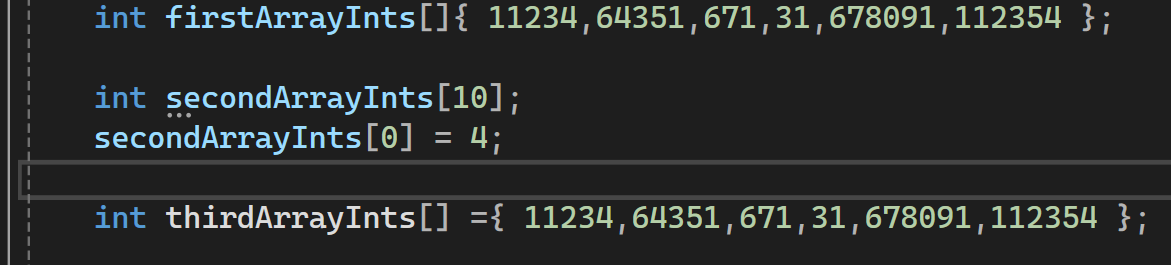
Break и Continue.

Работают идентично)

Массивы теория.

В целом логика работы массива таже самая, но есть небольшое НО, если мы попытаемся обратиться к массиву по индексу которого не существует в этом массиве, к примеру если у нас массив на 10 элементов, а мы попытаемся взять 100й, то есть вероятность того, что программа продолжит работу, а вместо наших данных мы получим какой-то мусор.

Инициализация массивов.

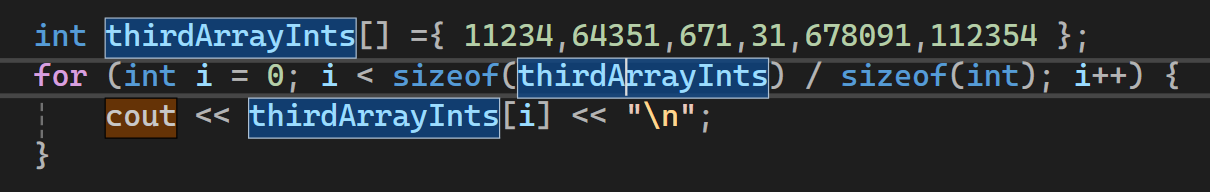


Примеры того, как можно инициализировать массив в c++

Ключевое слово sizeof для работы с массивами.

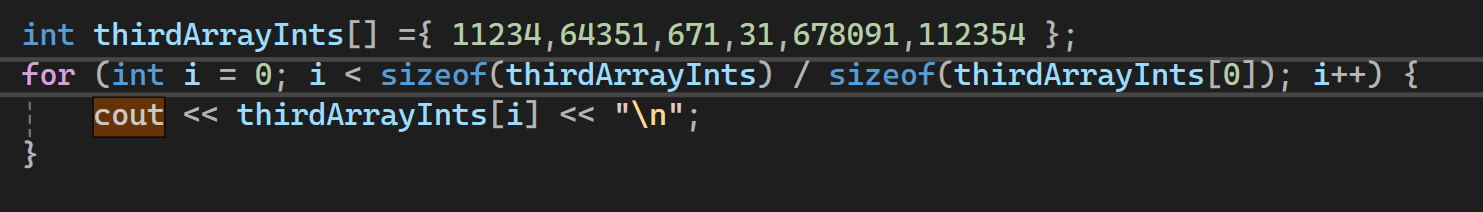
Sizeof(); - используется для вывода информации, сколько тот или иной тип данных занимает места в памяти, так-же с помощью этой функции мы можем узнать сколько та или иная переменная занимает места в памяти, и тоже самое можно сделать с массивами.

Если мы можем узнать сколько места в памяти занимает массив, и какой тип данных имеет этот массив, то с помощью этого мы легко можем узнать сколько в этом массиве элементов.



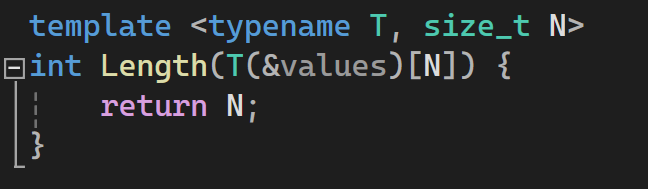
Пример кода, где в условии мы используем расчёт размера массива из известных данных.

Эта запись может быть не всегда удобна, так-как во втором sizeof мы берем конкретный тип данных, а что, если мы захотим его поменять? Тогда расчёты будут не правильные, исправим это.



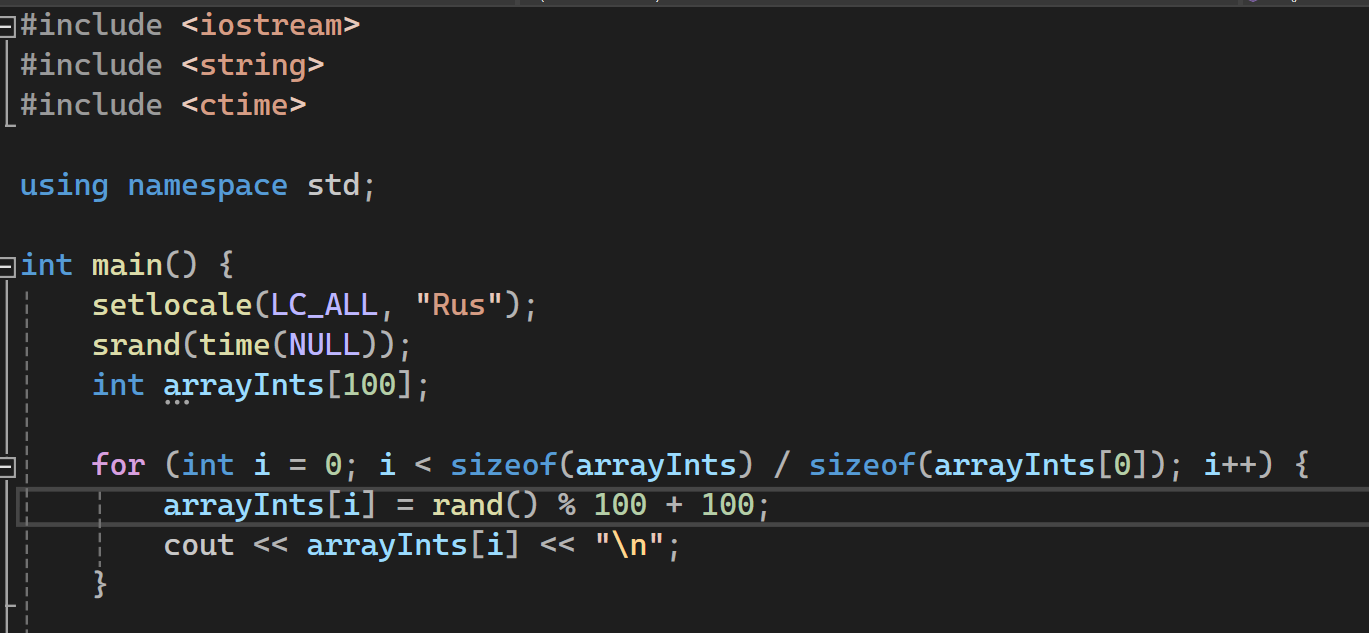
В этом случае мы берём нулевой элемент текущего массива, и если где-то он изменит свой тип, то расчёт будет правильный.

Пример функции, которая высчитывает размер массива. Как работает и что значит код смотри в Шаблонных функциях.



Генерация случайных чисел.

Для генерации случайных чисел используется следующий код.



Первым делом следует подключить библиотеку ctime, для использования времени, это необходимо для задания случайных чисел.

Дальше, нам нужно ввести число в функцию srand(); и в зависимости от того, какое число мы туда введём, у нас будут выводиться случайные числа, если у нас там окажется 2 одинаковых числа, то числа будут генерироваться по одинаковому принципу. Именно для этого мы используем время, так-как оно всегда разное и увеличивается.

После этого создаем массив на 100 элементов который мы хотим заполнить случайными числами.

В цикле, по длине массива мы проходим и заполняем его случайными числами из функции rand();

В примере используется не просто функция rand() а есть к ней приписка, эта приписка нужна для указания границ генерируемых чисел, %100 – означает что числа будут больше нижнего порога на 100, а +100 означает что нижний порог равен 100, получается нижний порог 100, а верхний больше на 100, и итоговые значения будут лежать от 100 до 200

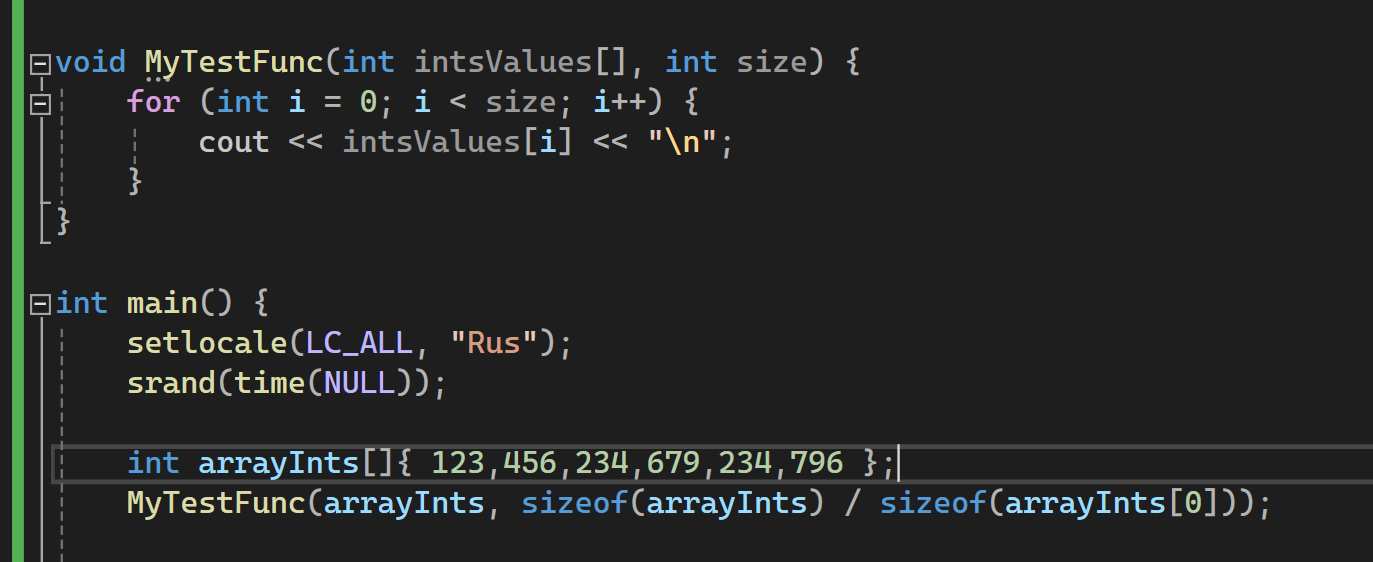
Функции.

Первое отличие — это то, что функция – это то, что находиться вне класса.

Функция должна быть объявлена до её вызова, если наоборот, сначала вызываем, а после, где-то ниже в коде объявляем то это работать не будет.

Передача параметров в функцию.

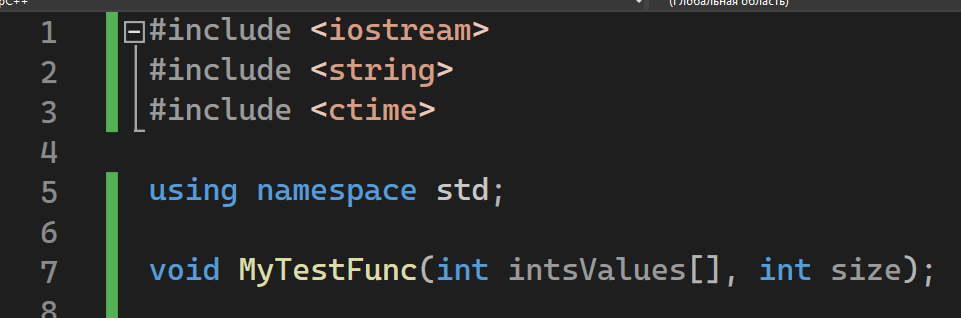
Передача значимых параметров в целом идентична c#, однако ссылочные данные ведут себя по-другому.

Если в шарпе мы получали ссылку на экземпляр, а под копотом он сам вставлял нужный экземпляр и мы работали с ним, то тут мы работаем с ссылкой, а значит мы не имеем доступа к данным, к примеру к размеру исходного массива, что заставляет нас передавать размер массива сразу. 

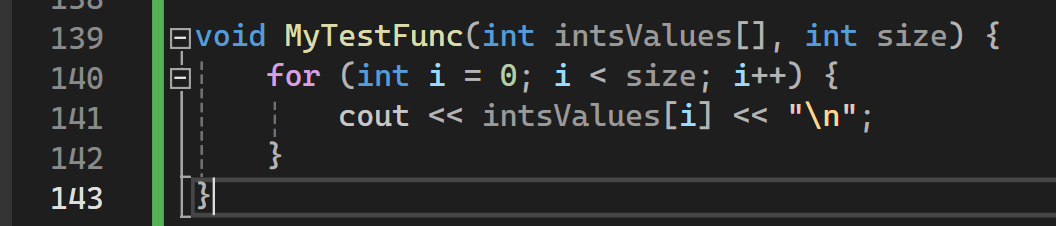
Прототипы функции.

Как писал выше, нельзя использовать функцию которая объявлена ниже, всё должно быть последовательно. Это не всегда удобно, особенно если ты пришёл с шарпа…

Для этого используются прототипы, нам нужно где-то сверху, до выполнения всех команд написать точно такую-же функцию, с такими-же входными параметрами и такими-же возвращаемыми значениями, только без блока кода, тоесть без фигурных скобок.

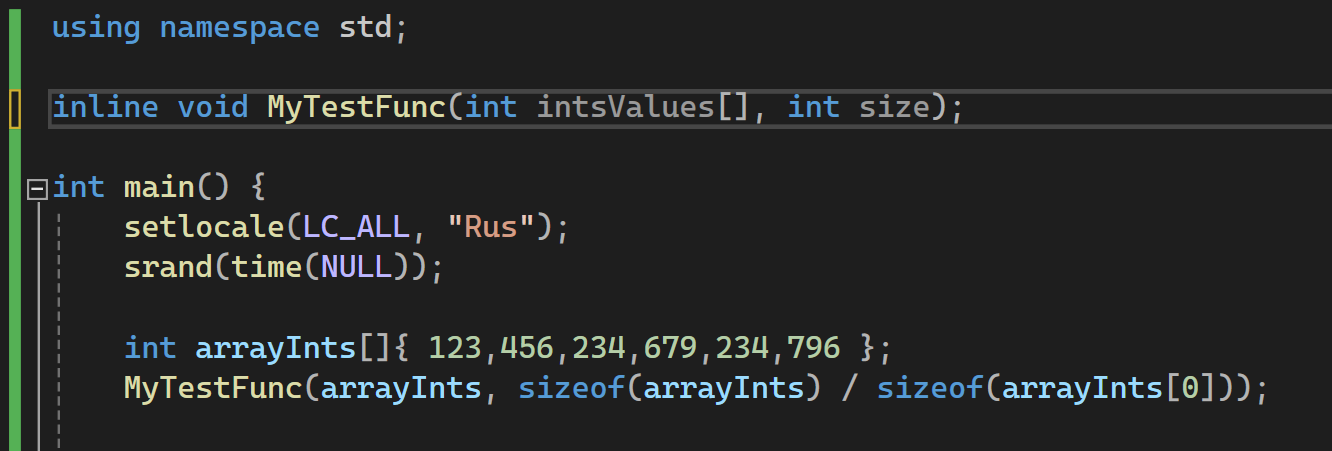


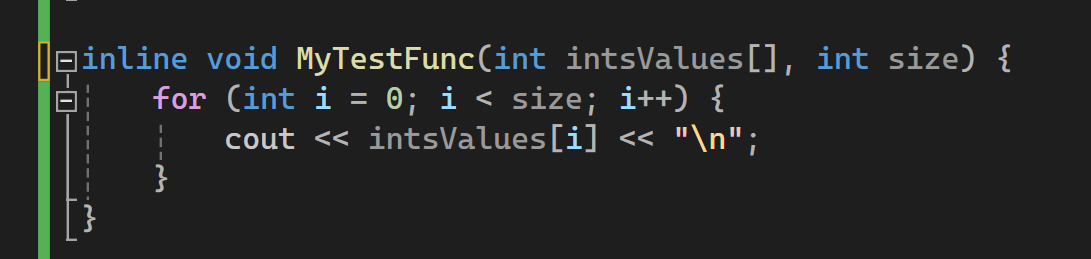
Функция прототип, которая просто говорит о том, что где-то есть такая функция, а вот основная функция с кодом находиться гораздо ниже



Ключевое слово Inline.

Раньше использовалось для оптимизации, но в последнее время умные компиляторы сами решают где оно нужно и где не нужно. При использовании этого слова с методом, вместо поиска этого метода в памяти, выполнения его содержимого и возврата, компилятор просто перенесёт всё содержимое в основной поток и выполнит там, а может и не выполнить, если функция большая, с множеством условий и ветвлений.

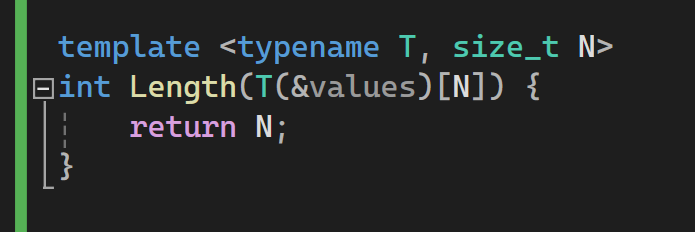




Шаблонные функции.

Чем-то похожи на шарповские дженерики, с их помощью можно работать с различными типами.

Как раз, та функция, которая возвращала длину массива и была шаблонной, рассмотрим теперь её подробнее.



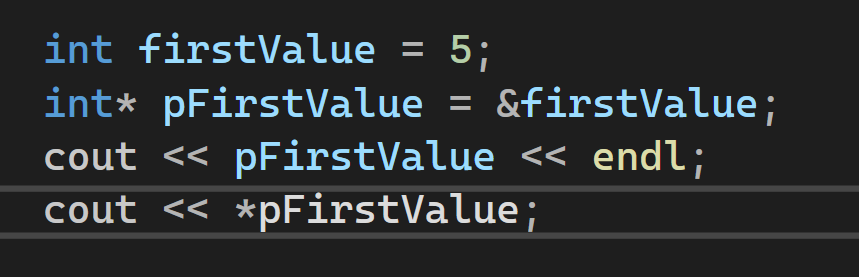
Первая строка как раз нам и говорит, что мы не знаем какой тип будет у входящих параметров и то, что мы хотим сделать тип шаблонный. Size\_t N говорит что мы можем принимать массивы любого размера, и из функции мы просто возвращаем N.

Также вместо слова typename можно использовать слово class. Разницы никакой, но лучше выбрать одну из вариаций и придерживаться её.

Указатели.

Указатель – это переменная, которая содержит в себе адрес другой переменной.

Пример использования и инициирования указателя.



Сначала создаем простую переменную

Дальше с помощью звёздочки, говорим что это УКАЗАТЕЛЬ типа ИНТ, а с помощью & - говорим что хотим присвоить в переменную pFirstValue – адрес в памяти нашей firstValue

Четвертой строкой мы получаем РЕАЛЬНЫЕ данные, которые лежат в переменной под адресом \*pFirstValue