<https://habr.com/ru/articles/646005/>

# Ссылки

T x;

T &rx = x; // rx это ссылка на x

Переменная x типа Т.

В переменной rx лежит адрес переменной x типа T.

Переменная rx становится псевдонимом переменной x.

Запись rx = 33; эквивалентна записи x = 33;.

# Ссылочный тип

Имя типа со спецификатором & будет называться ссылочным типом.

using RT = T&;

Можно использовать запись typedef T& RT;

Ссылочным типом можно заменить запись 1.2.

int x = 1

using RI = int&;

RI rx = x;

# Ссылки на массив

int a[4];

int(&ra)[4] = a;

Тип ссылки на массив включает размер массива, поэтому инициализировать нужно массивом того же размера.

При использовании псевдонима типа массива можно получить более привычный синтаксис определения ссылки на массив.

using I4 = int[4];

I4 a;

I4 &ra = a;

Можно объявить псевдоним ссылки на массив.

**using** RI4 = **int**(&)[4];

Доступ к элементу массива через ссылку осуществляется как обычно, с помощью индексатора.

int a[4];

int(&ra)[4] = a;

ra[0] = 42;

std::cout << ra[0];

# Передача массива констант в функцию

Определяем функцию

void showText1(const char str[])//функция принимает строку, как массив

{

cout << str << endl;

}

Передаем массив

showText1("~~~ ~~~ ~~~ cppstudio.com ~~~ ~~~ ~~~");

При компиляции программы компилятор для текстовой константы выделяет кусок памяти, равный длине константы.

При выполнении функции showText1 по адресу str сохраняется адрес ячейки, в которой лежит адрес первой ячейки выделенного для текстовой константы куска памяти.

Другими словами: при выполнении функции showText1 str содержит адрес указателя на выделенный для константы кусок памяти.

Указатель содержит в себе адрес первой ячейки памяти массива текстовой константы.

# Передача массива переменных в функцию

Определяем массив str1:

# char str1[] = "str1 - передаем, как массив в функцию void showText1 (char str[]);";

Для массива str1 компилятор выделил отдельный объём памяти, равный длине текстовой константы.

Передаём массив:

void showText1(const char str[])//функция принимает строку, как массив

{

cout << str << endl;

}

При выполнении функции showText1 по адресу str сохраняется адрес ячейки, в которой лежит адрес первой ячейки выделенного для str1 куска памяти.

Другими словами: при выполнении функции showText1 str содержит адрес указателя на выделенный для str1 кусок памяти.

При этом при выполнении функции str становится указателем на выделенный для str1 кусок памяти.

Указатель содержит в себе адрес первой ячейки памяти массива str1.

Копия массива не создаётся. Функция работает с данными, размещёнными в памяти, выделенной именно для str1 компилятором.

# Передаем в функцию указатель на массив переменных

Определяем массив str2:

# char str2[] = "str2 - передаем, в функцию void showText2 (char \*str); используя указатель.";

Для массива str2 компилятор выделил отдельный объём памяти, равный длине текстовой константы.

Передаём указатель на массив:

# void showText2(char\* str)//указатель \*str будет указывать на адрес первого символа в строке

# {

# cout << str << endl;

# }

При этом при выполнении функции по адресу str сохраняется адрес ячейки, в которой лежит адрес первой ячейки выделенного для str2 куска памяти.

Указатель содержит в себе адрес первой ячейки памяти массива str1.

Другими словами: при выполнении функции showText2 str содержит адрес указателя на выделенный для str2 кусок памяти.

Копия массива не создаётся. Функция работает с данными, размещёнными в памяти, выделенной именно для str2 компилятором.

# Передаём в функцию ссылку на строку символов

Ссылка – это адрес на область памяти с переменной или на первую ячейку массива переменных.

Определяем массив str3:

# char str3[150] = "str3 - передаем, в функцию void showText3 (char &str[]);\nТут используем адрес строки.";

Для массива str3 компилятор выделил отдельный объём памяти, равный длине текстовой константы.

Передаём ссылку на массив:

# void showText3(char(&str)[150])// адрес строки из 150-ти символов

# {

# cout << str << endl;

# }

Смысл заключённого в круглые скобки выражения: передаём в функцию showText3 массив символов, длиной 150, расположенный по адресу, имеющемуся в переменной str.

Действительно, адрес ячейка памяти, созданной для переменной str, оказался равен адресу первой ячейки массива str3. Так как длину массива str мы определили при вызове функции showText3, массив, доступный по переменной str является тем же самым массивом, созданным в памяти компилятором для переменной str3.

В данном случае str == str3, копия массива не создаётся, но переменная str становится псевдонимом переменной str3.

# Выводы по исследованию передачи массивов в функцию

Оптимизатор компилятора С++ не позволяет создавать в памяти копии массивов данных. Передача массива в функцию выполняется компилятором через создание указателя и массива ссылок на элементы исходного массива.

Если в функцию передаётся указатель на массив, то тем самым передается адрес первой ячейки массива и длина массива, чтобы компилятор мог создать массив ссылок на элементы передаваемого массива. Длину массива программист должен установить равным длине передаваемого в функцию массива.

Компилятор отслеживает совпадение длины массива, передаваемого в функцию, с длиной оригинала массива. Если длина передаваемого в функцию массива будет отличаться от длины массива-оригинала, компилятор выдаст ошибку о невозможности преобразовать переданный типа значения в заданный тип аргумента функции.

Если компилятор получает указатель на переменную, он автоматически через сохраненный в указателе адрес получает переменную указанного типа.

В случае получения указателя на массив компилятор по указателю получает только первый элемент массива. Однако для всего передаваемого массива компилятор создаёт массив ссылок на исходный массив. Изменяя значение адреса в указателе, можно получать ссылки на различные элементы массива.

В С++ можно проводить операции с адресом массива, записанным в указатель. Его можно инкрементировать, складывать со значением смещения, и т.д. После изменения значения адреса в указателе по указателю будет доступен элемент массива с соответствующим адресом.

# Ссылки

Для переменной в программе компилятор выделяет область памяти. Размер области памяти зависит от типа переменной.

Имя переменной в программе компилятор переводит на свой язык, понятный ему. Имя переменной компилятор понимает как адрес первой ячейки памяти, необходимой для размещения всех данных заданного типа.

Ссылка на переменную – это адрес первой ячейки области памяти, выделенной для переменной. Таким образом, компилятор и по имени переменной, и по имени ссылки, обратится к одной и той же области памяти, выделенной для этой переменной.

Инициировать ссылку можно знаком & в левой части выражения. Неважно, где будет стоять знак & - после типа переменной или перед именем переменной. Но если разместить знак & между типом и именем переменной, отделив от них пробелом, то некоторые редакторы убирают пробел между типом и знаком &.

Тип со знаком & в конце имени типа называется ссылочным типом.

Ссылка является псевдонимом переменной, поскольку работает с той же областью памяти, которая выделена для переменной.

При передаче ссылки на массив в функцию компилятор для выполнения функции создаёт массив ссылок на все элементы передаваемого массива.

# Указатели

Если ссылка по имени ссылочной переменной содержит непосредственно адрес переменной, на которую ссылка выполнена, то указатель – это отдельная переменная-посредник, которая хранит в себе адрес на ячейку, в которой хранится адрес переменной.

Указатель удобно использовать для выполнения стандартной операции над элементом массива. Изменил адрес в указателе – изменился адресуемый элемент в массиве.

Другой важный плюс указателя в том, что при передаче в функцию указателя на массив, не нужно устанавливать размер массива ссылок.

Отличие указателя от ссылки в том, что указатель является дополнительным элементом при доступе к ссылке на переменную. Вместо того, чтобы сразу получить доступ к ссылке, надо обратиться к указателю, в котором хранится ссылка на ссылку на переменную.