УВАЖАЕМЫЕ СТУДЕНТЫ! Изучите приведенную лекцию, законспектируйте основные понятия, дайте ответы на контрольные вопросы.

Ответы на вопросы, фотоотчет, предоставить преподавателю на e-mail: r.bigangel@gmail.com до 27.03.2023.

При возникновении вопросов по приведенному материалу обращаться по следующему номеру телефона: (072)111-37-59, (Viber, WhatsApp), vk.com: https://vk.com/daykini

ВНИМАНИЕ!!! При отправке работы, не забывайте указывать ФИО студента, наименование дисциплины, дата проведения занятия (по расписанию).

Лекция № 12

Тема: «Объявление и использование указателей в языке С++. Массивы в языке С++»

Цель: изучить способы объявления и использование указателей в языке С++, изучить массивы в языке С++.

План лекции:

1. ***Указатели***
2. ***Объявление указателей***
3. ***Массивы***

Указатель — это переменная, значением которой является адрес ячейки памяти. Указатели объявляются точно так же, как и обычные переменные, только со звёздочкой между типом данных и идентификатором:

```
int *iPtr; // указатель на значение типа int
double *dPtr; // указатель на значение типа double

int* iPtr3; // корректный синтаксис (допустимый, но не желательный)
int * iPtr4; // корректный синтаксис (не делайте так)

int *iPtr5, *iPtr6; // объявляем два указателя для переменных типа int
```

Синтаксически язык C++ принимает объявление указателя, когда звёздочка находится рядом с типом данных, с идентификатором или даже посередине. Обратите внимание, эта звёздочка не является оператором разыменования. Это всего лишь часть синтаксиса объявления указателя.

Однако, при объявлении нескольких указателей, звёздочка должна находиться возле каждого идентификатора. Это легко забыть, если вы привыкли указывать звёздочку возле типа данных, а не возле имени переменной. Например:

```
1 int* iPtr3, iPtr4; // iPtr3 - это указатель на значение типа int, а iPtr4 - это обычная переменная типа int!
```

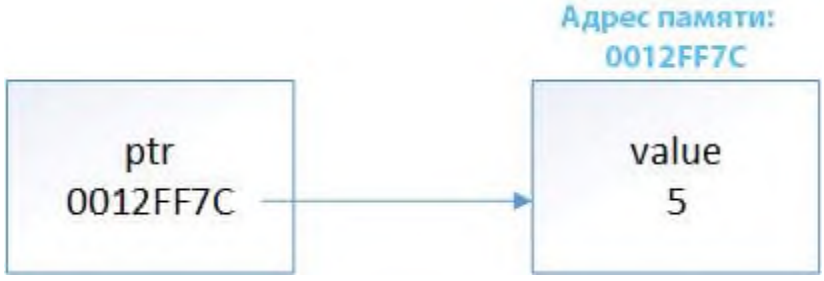
По этой причине, при объявлении указателя, рекомендуется указывать звёздочку возле имени переменной. Как и обычные переменные, указатели не инициализируются при объявлении. Содержимым неинициализированного указателя является обычный мусор.

Присваивание значений указателю

Поскольку указатели содержат только адреса, то при присваивании указателю значения — это значение должно быть адресом. Для получения адреса переменной используется оператор адреса:

```
1 int value = 5;
2 int *ptr = &value; // инициализируем ptr адресом значения переменной
```

Приведенное выше можно проиллюстрировать следующим образом:



Вот почему указатели имеют такое имя: ***ptr*** содержит адрес значения переменной ***value***, и, можно сказать, **ptr** указывает на это значение.

Еще очень часто можно увидеть следующее:

```
#include <iostream>

int main()
{
    int value = 5;
    int *ptr = &value; // инициализируем ptr адресом значения переменной

    std::cout << &value << '\n'; // выводим адрес значения переменной value
    std::cout << ptr << '\n'; // выводим адрес, который хранит ptr

    return 0;
}
```

Результат:

003AFCD4

003AFCD4

Тип указателя должен соответствовать типу переменной, на которую он указывает:

```
int iValue = 7;
double dValue = 9.0;

int *iPtr = &iValue; // ок
double *dPtr = &dValue; // ок
iPtr = &dValue; // неправильно: указатель типа int не может указывать на адрес переменной типа double
dPtr = &iValue; // неправильно: указатель типа double не может указывать на адрес переменной типа int
```

Следующее не является допустимым:

*int *ptr = 7*

Это связано с тем, что указатели могут содержать только адреса, а целочисленный литерал 7 не имеет адреса памяти. Если вы все же сделаете это, то компилятор сообщит вам, что он не может преобразовать целочисленное значение в целочисленный указатель.

Язык C++ также не позволит вам напрямую присваивать адреса памяти указателю:

*double *dPtr = 0x0012FF7C; // не ок: рассматривается как присваивание целочисленного литерала*

В чём польза указателей? Сейчас вы можете подумать, что указатели являются непрактичными и вообще ненужными. Зачем использовать указатель, если мы можем использовать исходную переменную?

Однако, оказывается, указатели полезны в следующих случаях:

Случай №1: Массивы реализованы с помощью указателей. Указатели могут использоваться для итерации по массиву.

Случай №2: Они являются единственным способом динамического выделения памяти в C++. Это, безусловно, самый распространенный вариант использования указателей.

Случай №3: Они могут использоваться для передачи большого количества данных в функцию без копирования этих данных.

Случай №4: Они могут использоваться для передачи одной функции в качестве параметра другой функции.

Случай №5: Они используются для достижения полиморфизма при работе с наследованием.

Случай №6: Они могут использоваться для представления одной структуры/класса в другой структуре/классе, формируя, таким образом, целые цепочки.

Указатели применяются во многих случаях. Не волнуйтесь, если вы многого не понимаете из вышесказанного. Теперь, когда мы разобрались с указателями на базовом уровне, мы можем начать углубляться в отдельные случаи, в которых они полезны, что мы и сделаем на последующих уроках.

Указатели — это переменные, которые содержат адреса памяти. Их можно разыменовать с помощью оператора разыменования * для извлечения значений, хранимых по адресу памяти. Разыменование указателя, значением которого является мусор, приведет к сбою в вашей программе.

Что такое массив?

К счастью, структуры не являются единственным агрегированным типом данных в языке C++. Есть еще массив — совокупный тип данных, который позволяет получить доступ ко всем переменным одного и того же типа данных через использование одного идентификатора.

Рассмотрим случай, когда нужно записать результаты тестов 30 студентов в классе. Без использования массива нам придется выделить 30 почти одинаковых переменных!

```
    // Выделяем 30 целочисленных переменных (каждая с разным именем)
    int testResultStudent1;
    int testResultStudent2;
    int testResultStudent3;
    // ...
    int testResultStudent30;
```

С использованием массива всё гораздо проще. Следующая строка эквивалентна коду, приведенному выше:

```
int testResult[30]; // выделяем 30 целочисленных переменных, используя фиксированный массив
```

В объявлении переменной массива мы используем квадратные скобки [], чтобы сообщить компилятору, что это переменная массива (а не обычная переменная), а в скобках — количество выделяемых элементов (это называется длиной или размером массива).

В примере, приведенном выше, мы объявили фиксированный массив с именем *testResult* и длиной 30. Фиксированный массив (или «массив фиксированной длины») представляет собой массив, размер которого известен во время компиляции. При создании *testResult*, компилятор выделит 30 целочисленных переменных.

Элементы массива

Каждая из переменных в массиве называется элементом. Элементы не имеют своих собственных уникальных имен. Вместо этого для доступа к ним используется имя массива вместе с оператором индекса [] и параметром, который называется индексом, и который сообщает компилятору, какой

элемент мы хотим выбрать. Этот процесс называется индексированием массива.

В вышеприведенном примере первым элементом в нашем массиве является testResult[0], второй — testResult[1], десятый — testResult[9], последний — testResult[29]. Хорошо, что уже не нужно отслеживать и помнить кучу разных (хоть и похожих) имен переменных — для доступа к разным элементам нужно изменять только индекс.

Важно: В отличие от повседневной жизни, отсчет в программировании и в языке C++ всегда начинается с 0, а не с 1!

В массиве длиной N элементы массива будут пронумерованы от 0 до N-1! Это называется диапазоном массива.

Пример программы с использованием массива

Здесь мы можем наблюдать как определение, так и индексирование массива:

```cpp
#include <iostream>

int main()
{
    int array[5]; // массив из пяти чисел
    array[0] = 3; // индекс первого элемента - 0 (нулевой элемент)
    array[1] = 2;
    array[2] = 4;
    array[3] = 8;
    array[4] = 12; // индекс последнего элемента - 4

    std::cout << "The array element with the smallest index has the value " << array[0] << "\n";
    std::cout << "The sum of the first 5 numbers is " << array[0] + array[1] + array[2] + array[3] + array[

    return 0;
}
```

Результат выполнения программы:

```
The array element with the smallest index has the value 3
The sum of the first 5 numbers is 29
```

Дополнительно ознакомиться:

https://ravesli.com/urok-81-nulevye-ukazateli/

https://ravesli.com/urok-82-ukazateli-i-massivy/

https://ravesli.com/urok-74-massivy-chast-1/

https://youtu.be/n0saIDd3H-M

https://youtu.be/Weh6UoLkNUQ

Контрольные вопросы:

1. Что такое указатель?
2. Как объявляется указатель?
3. Что такое массив?
4. Как объявляется массив?