What are Pointers?

In this lesson, we will explain the concept of pointers in C++. This will be crucial in our understanding of OOP.

Definition

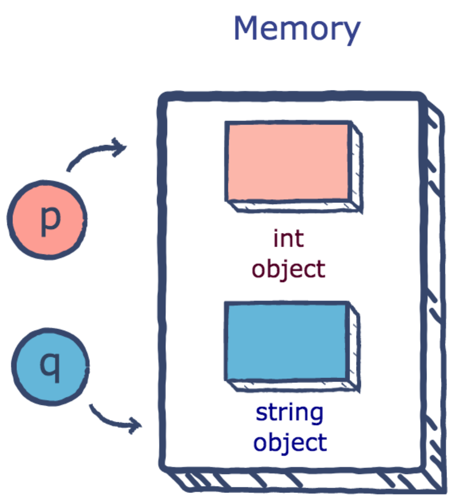
> A **pointer** holds the address of an object stored in memory. The pointer then simply "points" to the object.

The type of the object must correspond with the type of the pointer.

Declaration

A basic pointer can be declared using the following template:

```
type *name;    // points to a value of the specified type
```



`type` refers to the data type of the object our pointer points to, and `name` is just the label of the pointer. The `*` character specifies that this variable is in fact, a pointer. Here is an example:

```
int *p;    // integer pointer
string *q;    // string pointer
```

In the last lesson, we learned that the `&` character can be used to access the address of the variable succeeding it. This address can be assigned to a pointer!

The `*` character shown in the declaration above lets us access the value at the address. The process of accessing a pointer's value using `*` is also known as **dereferencing a pointer**.

Here's an interesting fact: arrays have been using pointers all this time! An array named `arr` can be treated as a pointer where the word `arr` is the name of a pointer and `arr[0]` corresponds to `*arr`.

```
#include <iostream>
using namespace std;

int main() {
  int var = 10;
  int *p;
  p = &var; // p points to the address of var
  cout << "The address of p: " << p << endl;
  cout << "The value of p: " << *p << endl;

  *p = 15; // update the value of p
  cout << "The new value of var is: " << var << endl; // var is updated!

  var = 20; // the value of var is updated
  cout << "The new value of *p and var: " << *p << endl; // p has been updated as well!

  int arr[] = {1, 3, 5, 7};
  p = arr; // p now points to the first element of arr
  cout << *p << endl; // the element at the zero-th index
  cout << *(p + 2) << endl; // the element at the second index
}
```

This is how a pointer deals with variables in static memory.

One must wonder about the point of all this. Why do we need pointers when we can simply manipulate the variables themselves?

We’ll answer this question in the next lesson.