Chapter 10: Pointers

A **pointer** stores the memory address of a variable.

```
int *p;
int* p;
```

To retrieve the memory address of a variable, use the **addressof operator** (&):

```
int v;
std::cout << &v; // prints the memory address of v
```

To access the data/value in the memory that the pointer points to, use the **dereferencing operator** (*):

```
int* p, v; // p is a pointer that stores a memory address of an integer, v is an integer variable
p = &v; // store the memory address of v to p
*p = 20; // set the data of the memory address to 20
```

Stack vs. Heap

Memory can be allocated on the **freestore**, or **heap**. Variables created on the heap are called **dynamically allocated variables**, or **dynamic variables**. Unlike variables declared on the stack (**automatic variables**), the programmer is responsible for managing the memory used in the heap.

Memory is allocated on the heap using the **new** keyword.

```
int* p = new int;
*p = 77;
```

Allocated memory is returned to the heap using the **delete** keyword. This is important to prevent **memory leaks**.

```
delete p;
```