Pointers

Terms

Address-of operator
Dereference operator
Free store
Heap memory
Indirection operator
Memory leak
Null pointer
Pointer
Reference parameters
Shared pointers
Smart pointers
Stack memory
Unique pointers

Summary

- Using the *address-of operator* (**&**) we can get the address of a variable.
- A *pointer* is a variable that holds the memory address of another variable.
- Using the *indirection* or *dereference operator* **(*)** we can get the content of a memory address stored in a pointer.
- One of the applications of pointers is to efficiently pass objects between function calls. Reference parameters are a safer and simpler alternative for the same purpose.
- A *null pointer* is a pointer that doesn't point to any objects.
- Local variables are stored in a part of memory called the *stack memory*. The memory allocated to these variables is automatically released when they go out of scope.

Copyright 2022 Code with Mosh

- We can use the **new** operator to dynamically allocate memory on a different part of memory called the *heap* (or *free store*).

- When allocating memory on the heap, we should always deallocate it using the **delete** operator. If we don't, our program's memory usage constantly increases. This is known as a *memory leak.*

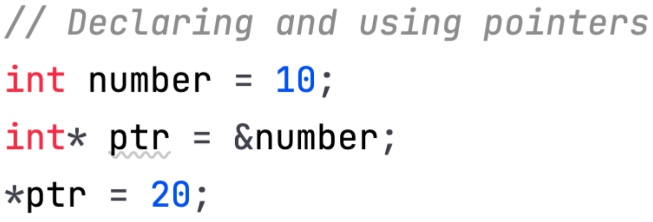
- *Smart pointers* in the STL are the preferred way to work with pointers because they take care of releasing the memory when they go out of scope.

- There are two types of smart pointers: *unique* and *shared*.

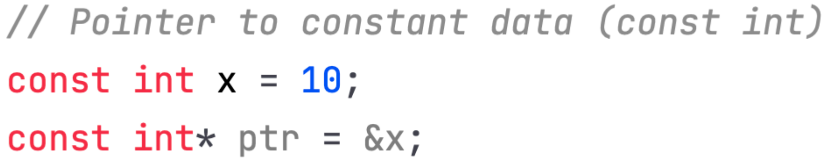
- A unique pointer owns the memory address it points to. So we cannot have two unique pointers pointing to the same memory location.

- If we need multiple pointers pointing to the same memory location, we have to use shared pointers.

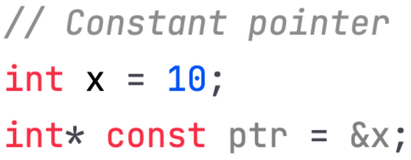
Copyright 2022 Code with Mosh



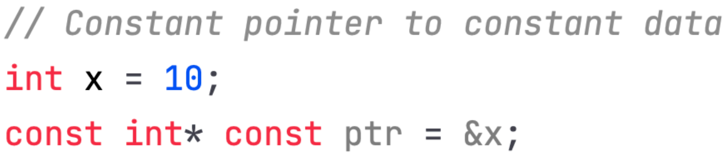
```
// Declaring and using pointers
int number = 10;
int* ptr = &number;
*ptr = 20;
```



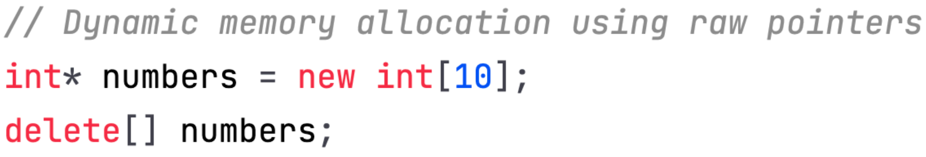
```
// Pointer to constant data (const int)
const int x = 10;
const int* ptr = &x;
```



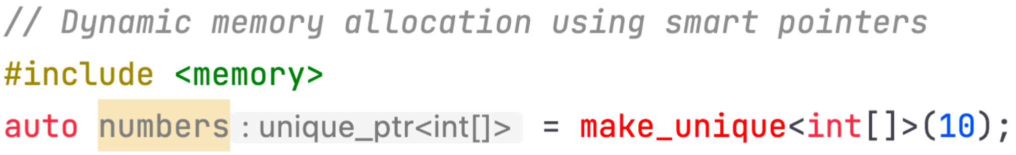
```
// Constant pointer
int x = 10;
int* const ptr = &x;
```



```
// Constant pointer to constant data
int x = 10;
const int* const ptr = &x;
```



```
// Dynamic memory allocation using raw pointers
int* numbers = new int[10];
delete[] numbers;
```



```
// Dynamic memory allocation using smart pointers
#include <memory>
auto numbers : unique_ptr<int[]>  = make_unique<int[]>(10);
```