Deallocation of Dynamic Memory

In this lesson, you will get acquainted with the deallocation of dynamic memory.

We'll cover the following

Introduction

The compiler automatically deallocates the static space when it is not used anymore. Since dynamically allocated memory is managed by a programmer, so when dynamically allocated space is not required anymore, we must free it.

`delete` operator

> *The **delete** operator allows us to free the dynamically allocated space.*

Syntax

The basic syntax for releasing the memory that the pointer is pointing to is given below:

delete pointer ;

Example program

See the program given below!

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

int main() {
```

```
int main() {
  // Declare pointer ptr
  int * ptr;

  // Store the starting address of dynamically reserved 4 bytes in ptr
  ptr = new int;
  // Store 100 in dynamic space
  *ptr = 100;
  // Print value pointed by ptr
  cout << *ptr;
  // Free the space pointed by pointer ptr
  delete ptr;
  return 0;
}
```

In the above program, the pointer is no longer pointing to the dynamic space that stores the value **100**.

One thing you might note here is that pointer `ptr` still exists in this example. So, we can reuse it later in the program to point to something else.

See the code given below!

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

int main() {
  // Declare pointer ptr
  int * ptr;
  // Store the starting address of dynamically reserved 4 bytes in ptr
  ptr = new int;
  // Store 100 in dynamic space
  *ptr = 100;
  // Print value pointed by ptr
  cout << *ptr << endl;
  // Free the space pointed by pointer ptr
  delete ptr;
  // Initialize a varible a
  int a = 70;
  // Store the address of a in ptr
  ptr = &a;
  // Prints the value pointed by the ptr
  cout << *ptr;
  return 0;
}
```

In the above code, initially, pointer `ptr` to the `int` value in the free store. We free the space pointed by the pointer `ptr`. After the deallocation, we store the address of variable `a` in pointer `ptr`. Now, pointer `ptr` points to the value of `a`.

> It’s good practice to set the pointer to `nullptr` after deallocation, unless you are pointing to some other valid target.

Quiz

The code given below will successfully compile.

```
double *ptr = new double;
  delete ptr;
  delete ptr;
```

Retake Quiz

Let’s study dynamic arrays in the upcoming lesson.