# Allocation of Dynamic Memory

Let's see how you can dynamically allocate the memory to the variables in C++.

**We'll cover the following**

## Introduction

For dynamic allocation, we have to do two steps:

- First, allocate the dynamic space.
- Then, store the starting address of the dynamic space in the pointer.

### `new` operator:

> *The unary operator **new** allocates memory in bytes during the run time from the free store.*

If memory is available on the free store, the new operator will reserve that memory and return its starting address.

### Syntax

The basic syntax for getting memory during the run-time is given below:

```
new  datatype ;
```

See the code given below!

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

int main() {
  // Reserve 4 bytes in free store
  new int;
  return 0;
}
```

In the above program, we are requesting **4 bytes** of memory from the free store. The `new` operator reserves the **4 bytes** of memory in the free store and returns the starting address of the allocated space. So, to access the dynamically allocated space, we must store the returned address somewhere. Here, pointers come in handy!

## Pointers #

To store the starting address returned by the new operator, we use pointers. The basic syntax is given below:

**datatype *pointer = new datatype**

```
#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;
  return 0;
}
```

**Line No. 8:** In the above program, we are storing the starting address of the allocated **4 bytes** in the pointer `ptr`.

## Accessing dynamic space #

Once we have reserved a dynamic space, how can we store something in that space?

Here, we use the dereferencing operator.

```
#include <iostream>
```

```
#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;
  return 0;
}
```

We can access the content of the dynamic space pointed by the pointer `ptr` using the dereference operator `*` before the pointer, and then store **100** in it.

Quiz

Q

What would the value of `*ptr1` at the end of this code be?

```
int * ptr1 , *ptr2;
  ptr1 = new int;
  ptr2 = new int;
  *ptr1 = 100;
  *ptr2 = 200;
  ptr1 = ptr2;
```

Retake Quiz

In the upcoming lesson, you will learn how to free dynamically allocated space.

Stay tuned!