Address-of Operator

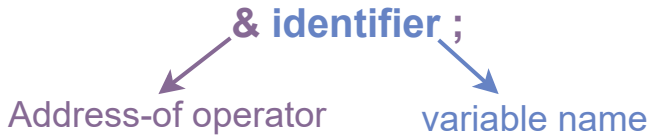
In this lesson, you will get acquainted with the address-of operator in C++.

We'll cover the following

Address-of operator

We now know that when we declare a variable, the compiler allocates space at someplace in the memory location. What if we want to know the memory address where the variable is allocated memory?

For this, we will use the address-of operator `&` before the identifier to access the address of the variable.



---

*The **address-of operator (&)** is a unary operator. It is used to extract the memory address of the variable.*

Example program

Consider the same analogy given in this lesson. Let's say `John`'s storage house is located somewhere in the **Memory Society**, and he has stored **10** in his house.

Let's write a C++ program to find the address of `John` house.

Press the **RUN** button and see the output!

```
#include <iostream>
```

```
using namespace std;

int main() {
  // Declare a variable John
  int John = 10;
  // Prints the memory address in which value of John is stored
  cout << "John Address = " << &John << endl;
  // Prints the value of John
  cout << "John Value = " << John << endl;
  return 0;
}
```

Explanation

Line No. 7: Stores 10 in `John`

Line No. 9: Shows us the address of `John`

> 📝 **0x** at the start of the memory address shows that they are in hexadecimal format.

Line No. 11: Displays the value stored in John

> **i** The **memory address** depends upon your machine. Therefore, if you run the same program on a different machine, you will get a different memory address.

We can access the variable address using the address-of (&) operator. Here, we have noticed that memory cells can store numbers, and addresses are numbers, too. So can we declare a variable and store the address of another variable in it?

Quiz

Q

What will the second statement in the following code snippet do?

```
int num = 10 , numAddress ;
```

```
numAddress = &num;
```

Retake Quiz

Let’s explore the functionality of pointers in the upcoming lesson.

Stay tuned!