C++ Variables Based on Scope

By:

Dr. Shweta Jain

Head Coordinator, Data Science

C++ Variables:

A variable is a name given to a memory location. It is the basic unit of storage in a program.

- The value stored in a variable can be changed during program execution.
- A variable is only a name given to a memory location, all the operations done on the variable effects that memory location.
- In C++, all the variables must be declared before use.

Variable Declaration:

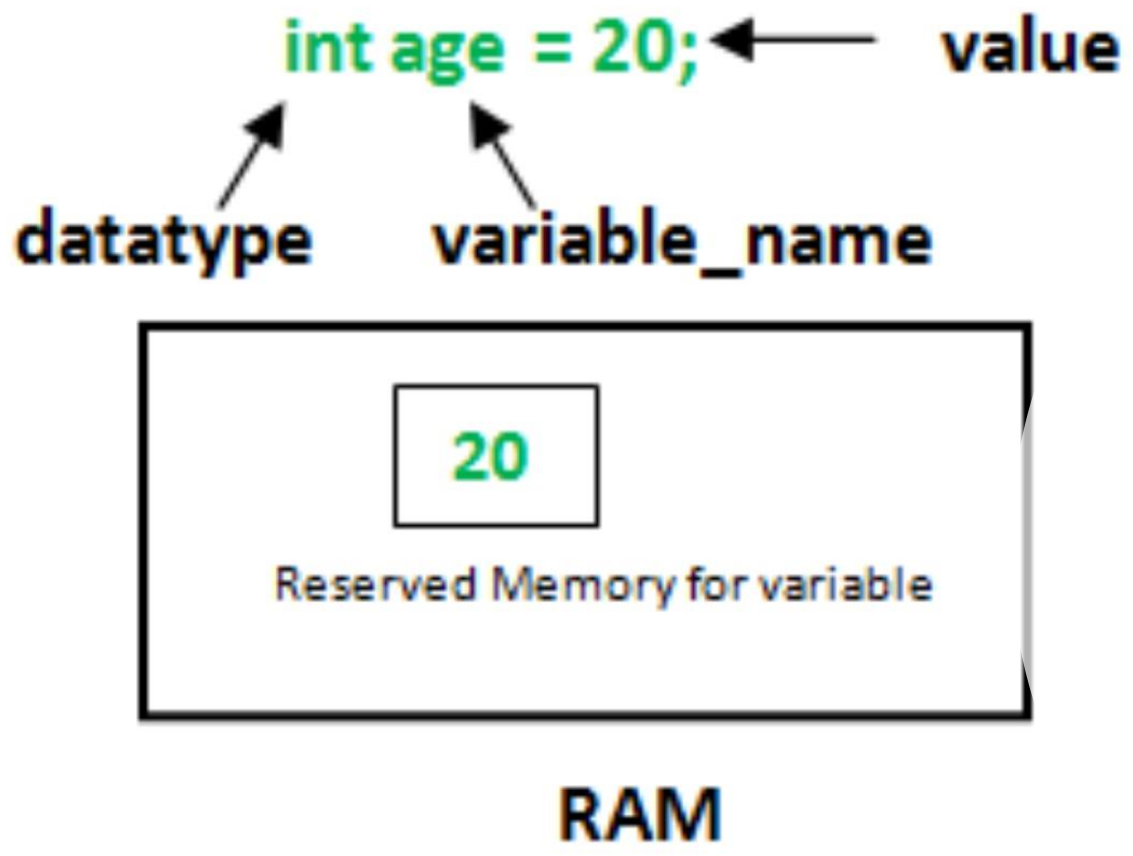
A typical variable declaration is of the form:

```
// Declaring a single variable
datatype variable_name;

// Declaring multiple variables:
datatype variable1_name, variable2_name, variable3_name;
```

Note: A variable name can consist of alphabets (both upper and lower case), numbers and the underscore ‘_’ character. However, the name must not start with a number.

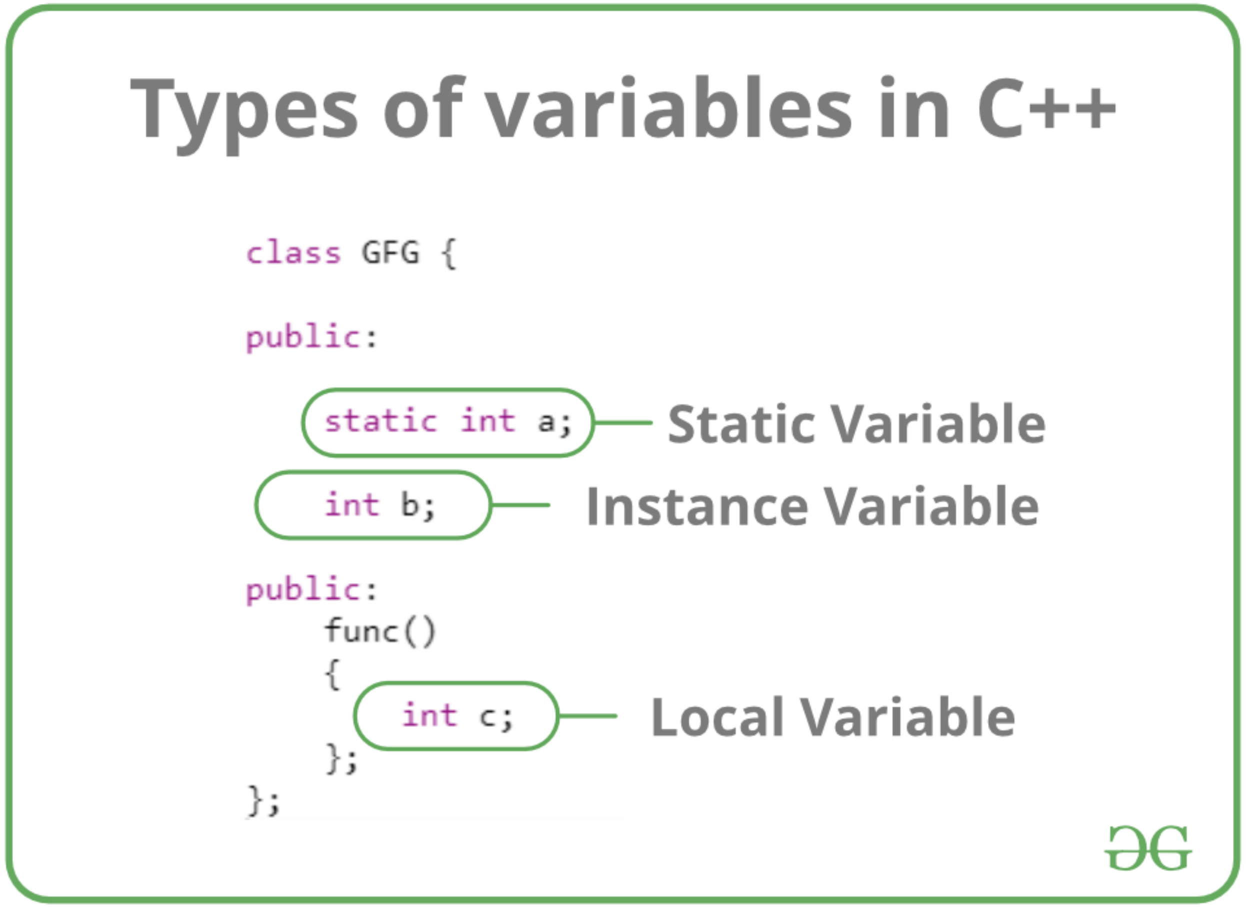
Variables in C++



Types of variables

There are three types of variables based on the scope of variables in C++:

1. Local Variables
2. Instance Variables
3. Static Variables



```
class GFG {

public:

    static int a;
  int b;

public:
    func()
    {
      int c;
    };
};
```

Local Variables:

- A variable **defined within a block** or method or constructor is called **local variable**.
- These variable are **created when the block in entered** or the function is called and **destroyed after exiting from the block** or when the call returns from the function.
- The **scope of these variables exists only within the block** in which the variable is declared. i.e. we can access these variable only within that block.
- **Initialisation** of Local Variable is **Mandatory**.

Instance Variables:

- Instance variables are basically the **data members of a class**.
- Instance variables **are non-static variables** and are declared in a class outside any method, constructor or block.
- As instance variables are declared in a class, these variables are **created when an object of the class is created** and **destroyed when the object is destroyed**.
- Unlike local variables, **we may use access specifiers for instance variables**. If we do not specify any access specifier then the default access specifier will be used.
- **Initialisation** of Instance Variable is **not Mandatory**.
- Instance Variable **can be accessed only by creating objects**.

Static Variables:

- Static variables are **also known as Class variables**.
- These variables are **declared similarly as instance variables**, the difference is that static variables are **declared using the static keyword within a class** outside any method constructor or block.
- Unlike instance variables, **we can only have one copy of a static variable per class** irrespective of how many objects we create.
- Static variables are **created at the start of program execution** and **destroyed automatically when execution ends**.
- **Initialization** of Static Variable is **not Mandatory**. Its default value is 0
- If we **access the static variable** like Instance variable (through an object), the compiler will show the warning message and it won't halt the program. The **compiler will replace the object name to class name automatically**.
- If we **access the static variable** without the class name, **Compiler will automatically append the class name**.

Instance variable Vs Static variable

- **Each object will have its own copy of instance variable** whereas We can **only have one copy of a static variable per class** irrespective of how many objects we create.
- **Changes made in an instance variable using one object will not be reflected in other objects** as each object has its own copy of instance variable. **In case of static, changes will be reflected in other objects** as static variables are common to all object of a class.
- We **can access instance variables through object references** and **Static Variables can be accessed directly using class name.**
- **Syntax** for static and instance variables:
- class Example:

```
{
   static int a; // static variable
   int b;        // instance variable
}
```