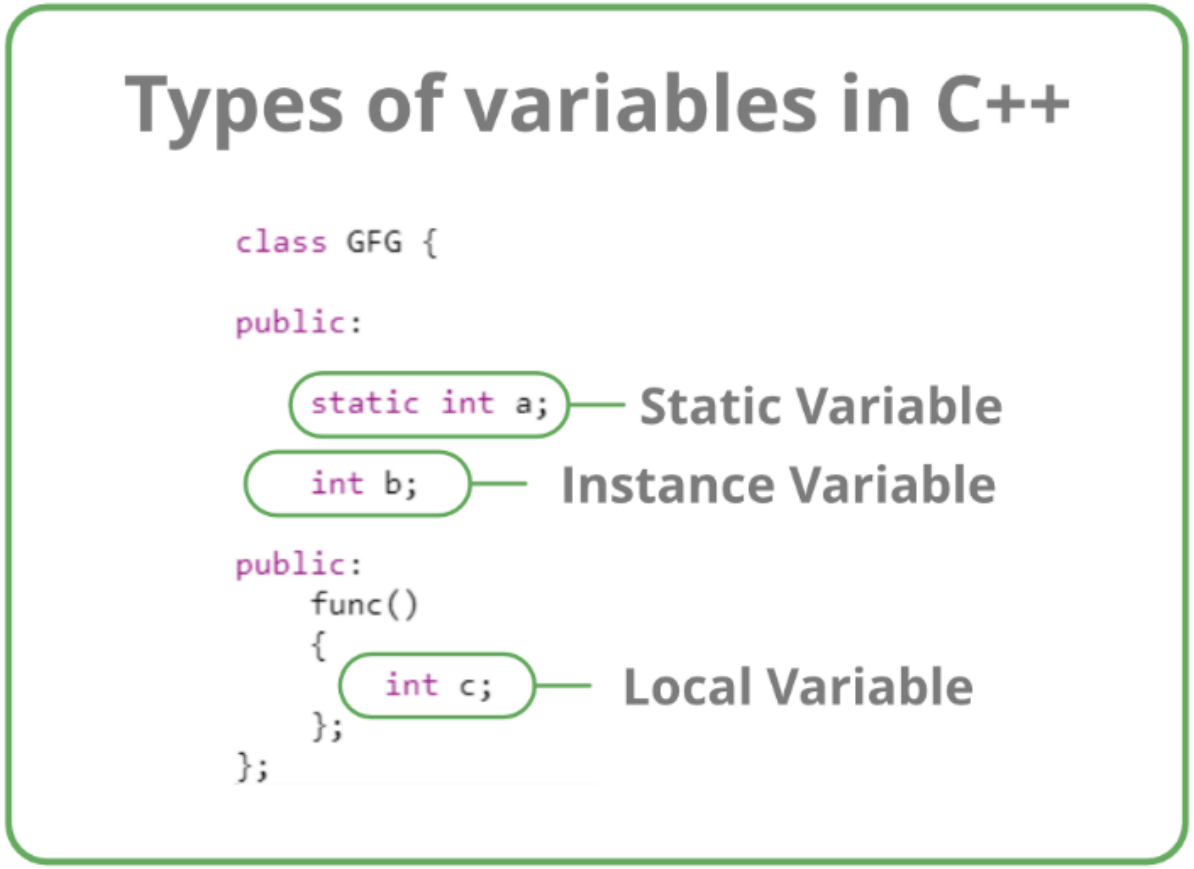
Variables & this Pointer in C++

Md. Alamgir Hossain
Senior Lecturer,
Dept. of CSE, Prime University
Mail: *alamgir.cse14.just@gmail.com*
YouTube: *https://www.youtube.com/alamgirhossaincse*
Blog: *https://alamgirhossainjust.blogspot.com/*

Types of Variables (Based on Scope of Variables)



Local Variable

- A variable defined within a block or method or constructor is called local variable.
- These variables are created when the ***block is 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 variables only within that block.
- ***Initialization of Local Variable is Mandatory.***

Instance Variable

- 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.
- ***Initialization of Instance Variable is not Mandatory.***
- Instance Variable can be accessed only by creating objects.

C++ Static Keyword

- In C++, ***static is a keyword or modifier*** that belongs to the type not instance.
- ***So instance is not required to access the static members.***
- In C++, static can be field, method, constructor, class, properties, operator, and event.
- **All the data members in a *static function* must be static.**

Advantages of C++ Static Keyword

- **Memory efficiency:** Now we don't need to create an instance for accessing the static members, so it saves memory.
- Moreover, it belongs to the type, so ***it will not get memory each time when the instance is created.***
- A field which is declared as static is called static field. Unlike instance field which gets memory each time whenever you create object, there is ***only one copy of static field created in the memory.*** It is shared to all the objects.

Example of Different Types Variable

```
1  #include<iostream>
2  using namespace std;
3  class prime{
4  public:
5      static int ID;//static variable
6      static string mobile;//static variable
7
8      int age;//Instance Variables
9      string location;//Instance Variables
10
11     void dusplay(){
12         int result = 3.50;//Local Variables
13         string name = "Mr.X";//Local Variables
14     }
15 };
16 int prime::ID;
17 int main()
18 {
19     cout<<prime::ID<<endl;//Static Variable Accessing
20     return 0;
21 }
```

C++ Static Function

```
1  #include<iostream>
2  using namespace std;
3  void display()
4  {
5      static int num1;
6      int num2 = 0;
7      num1++;
8      num2++;
9      cout<<num1<<" "<<num2<<endl;
10 }
11 int main()
12 {
13     display();
14     display();
15     return 0;
16 }
```

C++ Static Field

```cpp
#include <iostream>
using namespace std;
class student {
   public:
        int id;
        string name;
        static float result;//Static Variable Declaretion
        student(int roll, string s_name)
         {
              id = roll;
              name = s_name;
         }
        void display()
         {
             cout<<id++<<" "<<name<< " "<<result++<<endl;
         }
};
float student::result;//Definition of Static Variable
int main() {
    student s1 = student(10005, "Mr. X");
    student s2 = student(10006, "Mr. Y");
    s1.display();
    s1.display();
    s2.display();
    s2.display();
    return 0;
}
```

Value Initialization in C++ Static Variable

```cpp
#include<iostream>
using namespace std;
class p{
private:
    int a = 10;
    string s = "Prime";
    static float num;
public:
    void func()
    {
        num = 100;
    }
    void dispaly(){
        cout<<a<<" "<<s<<" "<<num++<<endl;
    }
};
float p::num;
int main()
{
    p obj, obj2, obj3;
    obj.func();
    obj.dispaly();
    obj2.dispaly();
    obj3.dispaly();
    return 0;
}
```

Static Member Function

```
#include<iostream>
using namespace std;
class p{
private:
    static int a;
    int b = 10;
    static float num;
public:
    static void func()
    {
        num = 100;
        a = 10;
    }
    void dispaly(){
        cout<<a++<<" "<<b++<<" "<<num++<<endl;
    }
};
float p::num;
int p::a;
int main()
{
    p obj, obj2, obj3;
    obj.func();
    obj.dispaly();
    obj2.dispaly();
    obj3.dispaly();
    return 0;
}
```

C++ this Pointer

In C++ programming, ***this is a keyword*** that refers to the current instance of the class. There can be 3 main usages of this keyword in C++.

- It can be used to ***pass the current object as a parameter to another method***.
- It can be used to ***refer current class instance variable***.
- It can be used to ***declare indexers.***

Example of C++ this Pointer

```cpp
#include <iostream>
using namespace std;
class Employee {
    public:
        int id; //data member (also instance variable)
        string name; //data member(also instance variable)
        float salary;
        Employee(int id, string name, float salary)
         {
              this->id = id;
              this->name = name;
              this->salary = salary;
         }
        void display()
         {
             cout<<id<<"  "<<name<<"  "<<salary<<endl;
         }
};
int main(void) {
    Employee e1 = Employee(101, "Sopno", 890000); //creating an object of Employee
    Employee e2 = Employee(102, "Sadhin", 59000); //creating an object of Employee
    e1.display();
    e2.display();
    return 0;
}
```

Thank You