**Objects Memory Allocation**

**Objects Memory Allocation in C++**

The way memory is allocated to variables and functions of the class is different even though they both are from the same class. The memory is only allocated to the variables of the class when the object is created.

The memory is not allocated to the variables when the class is declared. At the same time, single variables can have different values for different objects, so every object has an individual copy of all the variables of the class. But the memory is allocated to the function only once when the class is declared. So the objects don’t have individual copies of functions only one copy is shared among each object.

**Static Data Members in C++**

When a static data member is created, there is only a single copy of the data member which is shared between all the objects of the class. Usually, every object has an individual copy of the attributes unless specified statically.

Static members are not defined by any object of the class. They are exclusively defined outside any function using the scope resolution operator.

**An example of how static variables are defined is**

class Employee

{

public:

static int count; //returns number of employees

string eName;

void setName(string name)

{

eName = name;

count++;

}

};

int Employee::count = 0; //defining the value of count

**Static Methods in C++**

When a static method is created, they become independent of any object and class. Static methods can only access static data members and static methods. Static methods can only be accessed using the scope resolution operator. An example of how static methods are used in a program is shown.

#include <iostream>

using namespace std;

class Employee

{

public:

static int count; //static variable

string eName;

void setName(string name)

{

eName = name;

count++;

}

static int getCount()//static method

{

return count;

}

};

int Employee::count = 0; //defining the value of count

int main()

{

Employee Harry;

Harry.setName("Harry");

cout << Employee::getCount() << endl;

}