Object Oriented Programming in C++

Nepal College of Information Technology

Course Instructor: Er. Rabina Chaudhary

Memory Allocation:

- Static Memory Allocation
- Dynamic Memory Allocation

Static Memory Allocation: In static memory allocation, memory is allocated in stack and the memory allocated is automatically freed or de allocated by the compiler itself when the scope of the allocated memory finishes.

Example:

```
void main()
{
        int num;          // variable
        int arrayEg[5];  //  array
                              ...
}
```

```
class Student
      {
              ...
      };
      void main()
      {
              Student obj;     // object
              Student objArray[10]; //array of object
              ...
      }
```

Dynamic Memory Allocation:

- In dynamic memory allocation, the memory is allocated in heap
- While allocation of memory on heap, we need to delete the memory manually as memory is not de allocated or freed by the compiler itself even if the scope of allocated memory finishes
- We allocate and de allocate memory dynamically using **new** and **delete** operator respectively.

new Operator:

- The new operator is used for allocating memory dynamically
- new operator requests for the memory allocation in heap

Syntax to allocate memory using new operator:

```
pointer_variable = new datatype ;
```

Example:

```
int *ptr = new int;
```

new Operator:

- We can also initialize the memory using new operator.

Syntax:

```
pointer_variable = new datatype (value);
```

Example:

```
int *ptr = new int (50);
```

delete Operator:

- delete operator is used to de allocate the memory allocated by new operator

Syntax:

```
delete pointer_variable;
```

Syntax: To free dynamically allocated array pointed by pointer variable

```
delete [] pointer_variable:
```

Example Program:

```
 //Dynamically allocate memory  for an integer variable.
#include <iostream.h>
#include <conio.h>
            void main()
            {
                        int *ptr1=NULL;
                        ptr1  = new int ;     //declaring a integer variable dynamically
                        *ptr1 = 100;
                          int *ptr2 = new int (1234);

                        cout << "Value at address pointed by pointer variable 1 :"<< *ptr1 << endl;
                        cout << "Value at address pointed by pointer variable 2 :"<< *ptr2 << endl;
              delete ptr1;    //memory  pointed by ptr1 is released
              delete ptr2;    //memory  pointed by ptr2 is released

            getch();
            }
```

new Operator:

- Allocate block of memory using new operator:

  Syntax:

  ```
  pointer_variable = new datatype [ size ];
  ```

  Example:

  ```
  int *ptr = new int [10];
  ```

Delete Operator :

Syntax: To free dynamically allocated block of memory or array pointed by pointer variable

```
delete [] pointer_variable:
```

Example Program:

```
//Dynamically allocate memory for an integer array.
 #include <iostream.h>
 #include <conio.h>
void main()
{
                int *ptr=NULL;
                int n;
                cout << "Enter the size of array :";
                 cin >> n;
                ptr = new int [n];              //dynamically allocate memory for n integers
                 cout << endl << "Enter "<< n << " integer numbers :" << endl;
                 for(int i=0; i<n ; i++)
                {
                                cin >> *( ptr + i );
                }
                cout << endl << endl << endl ;
                 cout << "The values in array are : "<<endl;
                 for(int i=0; i<n ; i++)
                {
                                cout << *( ptr + i ) << endl;
                }
                delete [] ptr;     //deallocate memory pointed by ptr
                getch();
                }
```

```
//Dynamically allocating memory for object:Example Program
#include <iostream>
#include <conio>

class Student {
                         char name[50];
                          int roll;

   public:
                         void setInfo()
                           {
                         cout<< "Enter name of student : ";
                           cin >>name;
                           cout << "Enter roll of student : ";
                           cin >>roll;
                          }
                         void display()
                         {
                         cout << endl<<"Name= " << name << endl;
                          cout << endl<<"Roll= " << roll << endl;
                         }

};
```

```
//Program continued.


int main() {

         Student  *ptr1 = new Student;  // dynamically declare Student object
        cout<<endl<<"Enter  information  of student  :"<<endl;
        ptr1->setInfo();    // call setInfo() function

        cout<<endl<<"Information  of student  :"<<endl;
        ptr1->display();     // call display() function


         // pointed  memory  is released
          delete ptr1;

         getch();
         return  0;
}
```

```
//Dynamically allocating memory for array of objects: Example Program

#include<iostream.h>
#include<conio.h>

class Student {
                    int roll;
                   char name[50];
                    public:
                   void setInfo() {
                                        cout<<"Enter name :";
                                       cin>>name;
                                       cout<<"Enter roll number :";
                                       cin>>roll;
                                       cout<<endl<<endl;
                   }

                   void getInfo() {
                                        cout << "Name         = "<< name <<endl;
                                       cout << "Roll  number  = " << roll << endl<<endl;
                   }
};
```

```
//Program continued.

int main()
{
                         int n;
                        cout<<"Enter number of students :";
                        cin>>n;
                        Student* ptr = new Student[n];     // dynamically declare Student object array
                        for(int i=0;i<n;i++)
                        {
                                        (ptr+i)->setInfo();   // accessing class member through pointer, calling setInfo() function
                        }
                        cout<<endl<<endl<< "Information of Students are : "<<endl;
                        for(int i=0;i<n;i++)
                        {
                                        (ptr+i)->getInfo();       // accessing class member through pointer ,calling getInfo() function
                         }

                        delete []ptr;  // block of pointed memory is released

   getch();
   return 0;
}
```

Note:

- When new operator is used to allocate memory for a class’s object, the object’s constructor is called after the memory is allocated.
- When delete operator is used to deallocate memory for a class’s object, the object’s destructor is called before the objects memory is de allocated.

Example Program:

```
#include <iostream>
#include <conio>

class Test {
        int a,b;
        public:
                Test()
                {
                a=0;
                b=0;
                }
                Test(int x,int y)
                {
                a=x;
                b=y;
                }
void display()
        {
         cout << endl<<"a= " << a << endl;
        cout << endl<<"b= " << b << endl;
        }
  ~Test()
        {
        cout<<endl<<endl<<"Hey this is a
        destructor "<<endl<<endl;
        }
};
```

```
//Program Continued.

int main() {

            // dynamically declare Test objects
            Test *ptr1 = new Test;              // allocates memory for Test object and calls default constructor
            Test *ptr2 = new Test(100,200);    //allocates memory for Test object and calls parameterized consructor

            cout<<endl<<"Information of first object"<<endl;
            ptr1->display();
            cout<<endl<<"Information of second object"<<endl;
            ptr2->display();
  delete ptr1; //address pointed by ptr1 is released
  delete ptr2; //address pointed by ptr2 is released

  getch();
  return 0;
}
```