Academic Year: 2020

Semester: 2nd

Course Code: CS-241L

Course Title: Object Oriented Programming

CS-241L Object Oriented Programming Lab 03

Type of Lab: Open + Close Ended Weightage: 5%

CLO 2: Apply composition, inheritance and polymorphism and language extension concepts to build classes

Student understand the classes, constructors and dynamic memory allocation	**Cognitive/Understanding**	CLO2	Rubric A

Rubric A: Cognitive Domain

Evaluation Method: GA shall evaluate the students for Question according to following rubrics.

CLO	**0**	**1**	**2**	**3**	**4**	**5**
CLO2	Unable to understand and implement	Student understand the memory and classes concept	Implement dynamic memory allocation problem set	Completed all problem set of memory allocation	Completed half problem set of classes and constructors	Understood and implemented all the problem set of classes and memory concept

Lab 3a
BS-Computer Science Session
Object Oriented Programming

Target: Dynamically memory allocation, dynamic variables, new and delete operators

The process of allocating and deallocating memory space in a better way is called memory management. There are two ways that memory gets allocated for data storage:

1. Compile Time (or static) Allocation

- Memory for named variables is allocated by the compiler
- Exact size and type of storage must be known at compile time
- For standard array declarations, this is why the size has to be constant

2. Dynamic Memory Allocation

- Memory allocated "on the fly" during run time.
- dynamically allocated space usually placed in a program segment known as the heap or the free store
- Exact amount of space or number of items does not have to be known by the compiler in advance.
- For dynamic memory allocation, pointers are crucial

Important note:

- One use of dynamically allocated memory is to allocate memory of variable size which is not possible with compiler allocated memory except variable length arrays.
- The most important use is flexibility provided to programmers. We are free to allocate and deallocate memory whenever we need and whenever we don't need anymore. There are many cases where this flexibility helps. Examples of such cases are Linked List, Tree, etc.

How is it different from memory allocated to normal variables?

For normal variables like “int a”, “char str[10]”, etc. memory is automatically allocated and deallocated. For dynamically allocated memory like “int *p = new int[10]”, it is programmers’ responsibility to deallocate memory when no longer needed. If programmer doesn’t deallocate memory, it causes memory leak (memory is not deallocated until program terminates).

How is memory allocated/deallocated in C++?

C uses malloc() and calloc() function to allocate memory dynamically at run time and uses free() function to free dynamically allocated memory. C++ supports these functions and also has two operators new and delete that perform the task of allocating and freeing the memory in a better and easier way.

Differences between new operator and malloc() function in C++?

Both are used for same purpose, but still they have some differences, the differences are:

- new is an operator whereas malloc() is a library function.
- new allocates memory and calls constructor for object initialization. But malloc() allocates memory and does not call constructor.
- Return type of new is exact data type while malloc() returns void*.
- New is faster than malloc() because an operator is always faster than a Syntax to use new operator:

To allocate memory of any data type, the syntax is:

pointer-variable = new data-type;

Here, pointer-variable is the pointer of type data-type. Data-type could be any built-in data type including array or any user defined data types including structure and class.

```
// Pointer initialized with NULL

// Then request memory for the variable

int *p = NULL;

p = new int;

        OR

// Combine declaration of pointer

// and their assignment

int *p = new int;
```

We can also initialize the memory using new operator:

```
pointer-variable = new data-type(value);

Example:

int *p = new int(25);

float *q = new float(75.25);
```

Allocate block of memory:

new operator is also used to allocate a block(an array) of memory of type data-type.

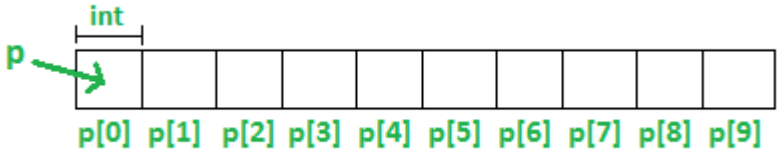
pointer-variable = new data-type[size];

where size(a variable) specifies the number of elements in an array.

Example:

```
int *p = new int[10]
```

Dynamically allocates memory for 10 integers continuously of type int and returns pointer to the first element of the sequence, which is assigned to p(a pointer). p[0] refers to first element, p[1] refers to second element and so on.



What if enough memory is not available during runtime?

If enough memory is not available in the heap to allocate, the new request indicates failure by throwing an exception of type std::bad_alloc, unless “nothrow” is used with the new operator, in which case it returns a NULL pointer (scroll to section “Exception handling of new operator” in this article). Therefore, it may be good idea to check for the pointer variable produced by new before using it program.

```
int *p = new(nothrow) int;
if (!p)
{
  cout << "Memory allocation failed\n";  }
```

delete operator:

Since it is programmer's responsibility to deallocate dynamically allocated memory, programmers are provided delete operator by C++ language.

Syntax:

// Release memory pointed by pointer-variable

delete pointer-variable;

Here, pointer-variable is the pointer that points to the data object created by new.

Examples:

delete p;

delete q;

To free the dynamically allocated array pointed by pointer-variable, use following form of delete:

// Release block of memory

// pointed by pointer-variable

delete[] pointer-variable;

Example:

```
// It will free the entire array pointed by p.

delete[] p;
```

Problem Set:

1. Write a program in which student array dynamically allocated and another marks array in which marks of students are stored. Find the average of students and highest marks of a student.

Hint: first check that memory is available or not for dynamically.

2. Write a C++ program in which user declare two-dimensional array dynamically. Find row sum and columns sum of all rows and columns of array.

Lab 3b

BS-Computer Science Session

Object Oriented Programming

Target: Define Simple Classes, Create Objects, Constructor

Class: It is a user defined data type, which holds its own data members and member functions.

The building block of C++ that leads to Object Oriented programming is a **Class**. For Example: Consider the Class of **Cars**. There may be many cars with different names and brand but all of them will share some common properties like all of them will have 4 wheels, Speed Limit, Mileage range etc. So here, Car is the class and wheels, speed limits, mileage are their properties.

- A Class is a user defined data-type which has data members and member functions.
- Data members are the data variables and member functions are the functions used to manipulate these variables and together these data members and member functions defines the properties and behaviour of the objects in a Class.
- In the above example of class Car, the data member will be speed limit, mileage etc. and member functions can be apply brakes, increase speed etc.

Basic Syntax of the class in C++

A class is defined in C++ using keyword class followed by the name of class. The body of class is defined inside the curly brackets and terminated by a semicolon at the end.

```
class ClassName
{
  Access Specifier:              //private, public or protected
  Data members;                  // Variables to be used
  Member Functions () {}         //Methods to access data members
};                               // Class end with a semicolon
```

Problem Set:

- Specify the components of the class from the below written code and explain the code.

```
class TestClass
{
   public:

   string name;

   void printname()
   {
     cout << "My name is: " << name;
   }
};
```

Object: An Object is an instance of a Class.

A class can be accessed and used by creating an instance of that class. A class is like a blueprint for an object. When a class is defined, no memory is allocated but when it is instantiated (i.e. an object is created) memory is allocated.

Object Declaration Syntax:

ClassName ObjectName;

Problem Set:

Dry run the code and write down the steps of execution to better understand the basic OOP program execution. Write the output of the program.

//C++ program to demonstrate the basic working of class & object

```
#include <stdio.h>
class Test
{
   // Access specifier
   public:

   // Data Members
   string name;

   // Member Functions()
   void printname()
   {
      cout << "My name is: " << name;
   }
};

int main() {

   // Declare an object of class Test
   Test obj1;

   // accessing data member
   obj1.name = "Mr. ABC";

   // accessing member function
   obj1.printname();
   return 0;
}
```

Constructor Calls: Constructors are special class members which are called by the compiler every time an object of that class is instantiated.

Constructors have the same name as the class and may be defined inside or outside the class definition. There are 3 types of constructors, which will be covered in detail, in the next lab. Today we'll explore default constructor.

Default Constructors: Default constructor is the constructor which doesn't take any argument. It has no parameters.

Problem Set:

With the help of comments in the code, explain it and modify it for 2 parameters in the constructor. Write down the output of the code.

```
#include <stdio.h>
class default_construct
{
   public:
   int id;

   //Default Constructor
   default_construct()
   {
      cout << "Default Constructor called" << endl;
      id=-1;
   }
};
int main() {

   // obj1 will call default Constructor
   default_constuct obj1();

   cout << "construct id is: " <<obj1.id << endl;
   return 0;
}
```

Problem Set:

- **Define a class batsman with the following specifications:**

Class of Batsman

Private Members	Type
Batcode (4 digit), Total_innings, n_out_innings, runs, bestscore	Integer
batavg	float
batname	10 character
Calavg() (Function to compute batsman avegerage)	float
Public Member:	
readdata() **detail**: Function to accept value from Batcode, name, innings, not_out and invoke the function Calcavg()	void
displaydata() **Detail:** Function to display the data members on the screen	Void

- **Define a class Student with the following specifications:**

Class of Student

Private Members	**Types**
Admission_num,	integer
Eng_m. math_m, science_m total _marks	float
S_name	10 character
ctotal() **Detail**: a function to calculate eng + math + science with float return type.	float
Public Member:	
Defualt Constructor() **Detail**: Function to accept values for admno, sname, eng, science. **Showdata()** **Detail:** Function to display all the data members on the screen and invoke **ctotal()** to calculate total marks.	

- **Define a class Flight with the following specifications:**

Class of Flights

Private Members	**Types**
Flight_num	Integer
Distance, Fuel_req	Float
S_name, Destination,	10 character
calfuel() **Detail:** A member function to calculate the value of Fuel as per the following **criteria:** if Distance <=1000 then Fuel 500 more than 1000 and <=1800 fuel 900 more than 1800 and less than 2200 1100 and more than 2200 then fuel 1300.	Float
Public Member:	
Default Constructor () **Detail**: to allow user to enter values for Flight Number, Destination, Distance. **Showdata()** **Detail:** Function to display all the data members on the screen & call function **calfuel()** to calculate the quantity of Fuel.	