Object Oriented Programming Laboratory

Assignment Sheet

Migrating C to C++

1. Write programs to understand to following concepts in C++. These programs need not get examined. Write these introductory programs to understand the basic concepts of C++.

   a) Flexible variable declaration
   b) C++ style of input/output
   c) C++ style type names
   d) Enumeration values
   e) Type casting
   f) The use of scope resolution operator (::)
   g) Constants:
   h) Constants and #define
   i) Constants and Pointers. (Also find some area of usage)
   j) References
   k) Constants and references
   l) Usage of references
   m) Functions that return by reference
   n) Inline function
   o) Inline function and macro
   p) Function overloading
   q) Default values of function parameters

C++ style Input/Output

2. Write a program that accepts two integers from keyboard, adds them and prints their values. Use cin and cout.
3. Create a factorial table using cout as follows:

   1! = 1

   2! = 2

   3! = 6

   …

   6! = 720

Flexible Variable Declaration

4. Write a program to print 1 to 10 using a for loop. Declare the loop variable inside the for loop. Check the scope of this variable.
5. Write a program to display Celsius to Fahrenheit conversion table using a for loop. Consider only 0˚ to 100˚ Celsius. Declare variables when they are used for the first time.

Constants

6. Write a program that defines a constant PI and takes radius of a circle from keyboard and prints area of that circle.
7. Write a function that takes an integer and returns the factorial of that number. Declare function parameter as const. Call the function with some argument from main function, store the result and print it.

Reference

8. Write a function swap() that takes two integer arguments and interchanges the values of those arguments using reference. Now in the main function, instantiate two integer variables with some values. Print their values. Call the swap function with these variables. Finally print the values of those variables. Check the result.
9. Now write another function swap() that takes two strings (character array) and interchanges them without reference parameters. Test this function using some arguments. Rewrite the function using reference parameters. Again test this function with some arguments.
10. Check if a swap function using reference takes more or less time than one using non-reference.

Constants and references

11. Write a function that takes an integer and returns the factorial of that number. Declare function parameter as read only reference. Call the function with some argument from main function, store the result and print it.

Constants and pointers

12. Write a function Strcpy to copy one string to another with suitable formal parameters declarations. Following points must be considered.
    a) Source string must not get modified
    b) Target string is allowed to get modified
    c) The Pointers must be constant pointers.

    Use it to copy some strings.

Inline function

13. Write an inline function add() that takes three integer arguments and returns the sum of these arguments.
14. Check if the above inline add function takes more or less time than a non-inline version of add function.

Function Overloading

15. Consider the following two scenarios:
    a) We want to find out the maximum between three integers.
    b) We also want to find out the maximum element of an array of integers.

    Write two overloaded functions for these two scenarios.
16. Write two overloaded functions print() such that one prints the elements of a vector and the other prints elements of a matrix. Note that a vector and a matrix may be represented as a one-dimensional array and a two-dimensional array respectively.

Default values for function parameters

17. Consider function add() in 3. Specify the default values for second and third parameters to 0 (zero). Now call this function with three, two and one arguments and see the result.

Classes and Objects

18. Write a simple class that represents a class of geometrical points each of which has three coordinates. The class should have appropriate constructor(s). Also add a m:!ember function distance() that calculates Euclidian distance between two points. Now create two points, find the distance between them and print it.
19. Write a class for the geometrical shape rectangle. Write suitable constructors and member functions. Add a member function area() that calculates the area of a rectangle. Create 4 rectangles and print their respective area.
20. Write a class that represents a class of wireless device. A device has a location (point object may be used), a fixed unique id, and a fixed circular transmission range. Write suitable constructors and member functions for this class. Instantiates 10 such devices. Choose location (coordinates) and transmission range of the devices randomly. Now, for each of these devices, find the neighbor devices (i.e. devices that belong to the transmission range). Suppose, all of these devices have moved to a new location (randomly chosen). Find out the new set of neighbors for each of these devices.
21. Write a class Vector for one dimensional array. Write suitable constructor/copy constructor. Also add member functions for perform basic operations (such as addition, subtraction, equality, less, greater etc.). Create vectors and check if those operations are working correctly.
22. Write a class IntArray for one dimensional integer array. Implement the necessary constructor, copy constructor, and destructor (if necessary) in this class. Implement other member functions to perform operations, such adding two arrays, reversing an array, sorting an array etc. Create an IntArray object having elements 1, 2 and 3 in it. Print its elements. Now, create another IntArray object which is an exact copy of the previous object. Print its elements. Now, reverse the elements of the last object. Finally print elements of both the objects.
23. Create a simple class SavingsAccount for savings account used in banks as follows: Each SavingsAccount object should have three data members to store the account holder's name, unique account number and balance of the account. Assume account numbers are integers and generated sequentially. Note that once an account number is allocated to an account, it does not change during the entire operational period of the account. The bank also specifies a rate of interest for all savings accounts created. Write relevant methods (such as withdraw, deposit etc.) in the class. The bank restricts that each account must have a minimum balance of Rs. 1000. Now create 100 SavingsAccount objects specifying balance at random ranging from Rs. 1,000 to 1,00,000. Now, calculate the interest for one year to be paid to each account and deposit the interest to the corresponding balance. Also find out total amount of interest to be paid to all accounts in one year.
24. Write some programs to understand the notion of constant member functions, mutable data members etc.
25. Write the definition for a class called Complex that has private floating point data members for storing real and imaginary parts. The class has the following public member functions:

    `setReal()` and `setImg()` to set the real and imaginary part respectively.

    `getReal()` and `getImg()` to get the real and imaginary part respectively.

    `disp()` to display complex number object

    `sum()` to sum two complex numbers & return a complex number

    Write main function to create three complex number objects. Set the value in two objects and call sum() to calculate sum and assign it in third object. Display all complex numbers.
26. Complete the class with all function definitions for a stack

```
class Stack {
        int *buffer, top;
public :
        Stack(int);                 //create a stack with specified size
        void push(int);  //push the specified item
          int pop();                        //return the top element
               void disp();                 //displays elements in the stack in top to bottom order
```

}

Now, create a stack with size 10, push 2, 3, 4 and 5 in that order and finally pop one element. Display elements present in the stack.

27. Complete the class with all function definitions for a circular queue

```
class Queue {
        int *data;
        int front, rear;
public :
        Queue(int );      //create queue with specified size
        void add(int);    //add specified element to the queue
        int remove();//delete element from the queue
                void disp();          //displays all elements in the queue(front to rear order)
}
```

Now, create a queue with size 10 add 2, 3, 4 and 5 in that order and finally delete two elements. Display elements present in the stack.

28. Write a class for your Grade card. The grade card is given to each student of a department per semester. The grade card typically contains the name of the department, name of the student, roll number, semester, a list of subjects with their marks and a calculated CGPA. Create 60 such grade cards in a 3rd semester with relevant data and find the name and roll number of student having highest CGPA.
29. Create a class for Book. A book has unique isbn (string), title, a list of authors, and a price. Write relevant functions. Now write a class BookStore which has a list of books. There may be multiple copies of a book in the book store. Write relevant member functions. Write a function books() that returns list of unique isbn numbers of the books, a function noOfCopies() that returns the number of copies available for a given isbn number and a function totalPrice() that returns the total price of all the books. Create a book store having a number of books (multiple copies). Now, for each book, print number of copies of that book along with its title.

Operator Overloading

30. Write a class "**Point**" which stores coordinates in (x, y) form. Define necessary constructor, destructor and other reader/writer functions. Now overload '-' operator to calculate the distance between two points.
31. Design a class ***Complex*** that includes all the necessary functions and operators like =, +, -, *, /.
32. Implement a class "**Quadratic**" that represents second-degree polynomial i.e. polynomial of type ax^2+bx+c. The class will require three data members corresponding to a, b and c. Implement the following:
    a) A constructor (including a default constructor which create a null polynomial)
    b) Overload the addition operator to add two polynomials of degree 2.
    c) Overload << and >> operators to print and read polynomials.
    d) A function to compute the value of polynomial for a given x.
    e) A function to compute roots of the equation $ax^2+bx+c=0$. Remember, root may be a complex number. You may implement "**Complex**" class to represent root of the quadratic equation.
33. A program is given as follows:

```
class INT {
        int i;
    public :
        INT(int a):i(a){}
        ~INT() {}
};
int main() {
    int x = 3;
    INT y = x;
```

```
    y++ = ++y;
    x = y;
    return 0;
}
```

Write extra functions/operators required in the INT class to make main program work. Provide suitable implementation for the added functions/operators.

34. Design and implement class(es) to support the following main program.

```
int main() {
    IntArray i(10);
    for(int k = 0; k < 10; k++)
        i[k] = k;
    cout << i;
    return 0;
}
```

35. You are given a main program:

```
int main() {
    Integer a = 4, b = a, c;
    c = a+b++;
    int i = a;
    cout << a << b << c;
    return 0;
}
```

Design and implement class(es) to support the main program.

36. Design and implement class(es) to support the following code segment.

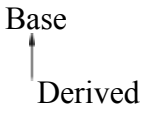
```
Table t(4, 5), t1(4, 5);
cin >> t;
t[0][0] = 5;
int x = t[2][3];
t1 = t;
cout << t << "\n" << t1;
```

37. Design and implement class(es) to support the following code segment.

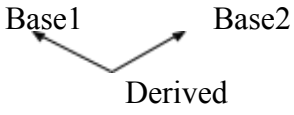
```
Index in(4), out(10);
int x = in;
int y = in + out;
in = 2;
Integer i;
i = in;
```

Inheritance

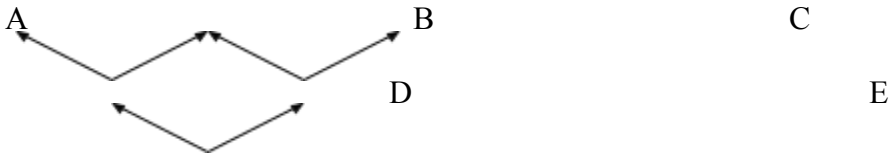
38. Write empty class declarations for the following class hierarchy.



39. Write empty class declarations for the following class hierarchy.



40. Write empty class declarations for the following class hierarchy.



41. Write a class Person having data member name, age, height etc. Write proper constructors, methods to get/set them and a method printDetails() that prints all information of a person. Now write another class Student from Person and add data members roll, year of admission etc. Write constructors, methods to get/set them and a override printDetails(). Now create a Person and a Student object and call printDetails() function on them to display their information.

    Now Create an array of pointers to Person and store addresses of two Persons and two Students. Call printDetails() on all elements (a loop may be used). Are you getting output which is supposed to come? Make printDetails() function virtual in the base class and check the result.

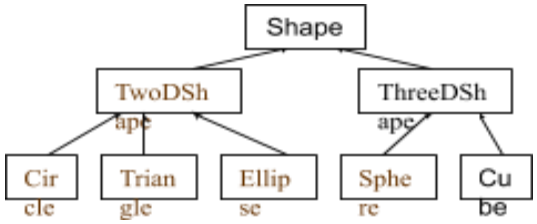
42. Write a class Employee having data member name, salary etc. Write proper constructors, methods to get/set them and a virtual method printDetails() that prints all information of a person. Now write two classes Manager and Clerk from Employee. Add 'type' and 'allowance' in the manager and Clerk respectively. Write constructors, methods to get/set them and a override printDetails(). Now create a Manager and a Clerk object and call printDetails() function on them to display their information.

    Now Create an array of pointers to Employee and store addresses of two Employee, two Managers and two Clerks. Call printDetails() on all elements (a loop may be used). Also find the total salary drawn by all employees.

43. Write class definitions for the following class hierarchy

    The Shape2D class represents two dimensional shapes that should have pure virtual functions area(), perimeter() etc. Implement these functions in Circle and Rectangle. Also write proper constructor(s) and other functions you think appropriate in the Circle and Rectangle class. Now create an array of 5 Shape2D pointers. Create 3 Circle and 2 Rectangles objects and place their addresses in that array. Use a loop to print area and perimeter of all shapes on this array.

44. ***Implement*** the Shape hierarchy as shown in the figure. Each ***TwoDShape*** should contain function ***getArea*** to calculate the area of two-dimensional shape. Each ***ThreeDShape*** should have member functions ***getArea*** and ***getVolume*** to calculate the surface area and volume of the three-dimensional shape respectively. Create a program that uses Vector of Shape pointers to objects of each concrete class in the hierarchy. Now write a program that processes all the shapes in the Vector such that if the shape is a ***TwoDShape*** it prints name of shape and its area while it prints name of shape, its area and volume if the shape is a ***ThreeDShape***.



45. Write a program to illustrate the role of virtual destructor.

Exception Handling

46. Two integers are taken from keyboard. Then perform division operation. Write a try block to throw an exception when division by zero occurs and appropriate catch block to handle the exception thrown.

47. Design a class Stack with necessary exception handling.

48. Vehicles may be either stopped of running in a lane. If two vehicles are running in opposite direction in a single lane there is a chance of collision. Write a C++ program using exception handling to avoid collisions. You are free to make necessary assumptions.

Templates

49. Write a template function max() that is capable of finding maximum of two things (that can be compared). Used this function to find (i) maximum of two integers, (ii) maximum of two complex numbers (previous code may be reused). Now write a specialized template function for strings (i.e. char *). Also find the maximum of two strings using this template function.
50. Write a template function swap() that is capable of interchanging the values of two variables. Used this function to swap (i) two integers, (ii) two complex numbers (previous code may be reused). Now write a specialized template function for the class Stack (previous code may be reused). Also swap two stacks using this template function.
51. Create a C++ template class for implementation of Stack data structure. Create a Stack of integers and a Stack of complex numbers created earlier (code may be reused). Perform some push and pop operations on these stacks. Finally print the elements remained in those stacks.