Department of CSE

Object Oriented Programming using C++(ACCS-16302)

Solution

Section A

Q1)

i. Call by value method copies the value of an argument into the formal parameter of that function. Call by reference method copies the address of an argument into the formal parameter. In call by value method, a copy of the variable is passed whereas, In call by reference method, a variable itself is passed.

ii. Function templates are special functions that can operate with generic types. This allows us to create a function template whose functionality can be adapted to more than one type or class without repeating the entire code for each type.

iii. C++ templates enable you to define a family of functions or classes that can operate on different types of information. Use templates in situations that result in duplication of the same code for multiple types

iv. Code block,Turbo c++,Gcc++,Dev c++,Borland C++

v. endl. The endl manipulator works the same way as the '\n' character in C++. That is the endl manipulator outputs the subsequent data or text in the next line. ...
setw manipulator. This manipulator is used to set the width of the output in a program.

vi. Data hiding is a technique especially practised in object-oriented programming (OOP).Data hiding is hiding the details of internal data members of an object.

vii. A friend function of a class is defined outside that class' scope but it has the right to access all private and protected members of the class.

viii. A static member function can only access static data member, other static member functions and any other functions from outside the class.

ix. Static member functions have a class scope and they do not have access to the this pointer of the class. You could use a static member function to determine whether some objects of the class have been created or not.

x. Infinity loop

Section B

Q2) Yes, C++ is an Object Oriented Programming language .The various features of object oriented:

- Inheritance
- Polymorphism
- Data Hiding
- Encapsulation
- Overloading

- Reusability

Q3 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 behavior of the objects in a Class.

Difference between object and class

No.	Object	Class
1)	Object is an instance of a class.	Class is a bluepr or template fr which objects created.
2)	Object is a real world entity such as pen, laptop, mobile, bed, keyboard, mouse, chair etc.	Class is a group similar objects.
3)	Object is a physical entity.	Class a logical entity.
4)	Object is created through new keyword mainly e.g. Student s1=new Student();	Class is dec using class keyword e.g. class Student{}
5)	Object is created many times as per requirement.	Class declared once.
6)	Object allocates memory when it is created.	Class doesn't allocated me when it is create
7)	There are many ways to create object in java such as new keyword, newInstance() method, clone() method, factory method and deserialization.	There is only on to define class i using class keyw

```
using namespace std;
class Geeks
```

```
{
  // Access specifier
  public:

  // Data Members
  string geekname;

  // Member Functions()
  void printname()
  {
    cout << "Geekname is: " << geekname;
  }
};

int main() {

  // Declare an object of class geeks
  Geeks obj1;

  // accessing data member
  obj1.geekname = "Abhi";

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

Q4
```
#include <stdio.h>
#include <string.h>

int main(int argc, char * argv[])
{
const char * filename="test.txt";
const char * mytext="Once upon a time there were three bears.";
int byteswritten=0;
FILE * fl= fopen(filename, "wb") ;
if (fl) {
 fwrite(mytext,sizeof(char),strlen(mytext), fl) ;
 fclose( fl ) ;
}
printf("len of mytext = %i ",strlen(mytext)) ;
return 0;
}
```

Q5)

Dangling pointer

A pointer pointing to a memory location that has been deleted (or freed) is called dangling pointer. There are three different ways where Pointer acts as dangling pointer

```
int main()
```

```
{
  int *ptr = (int *)malloc(sizeof(int));

  // After below free call, ptr becomes a
  // dangling pointer
  free(ptr);

  // No more a dangling pointer
  ptr = NULL;
}
```

NULL Pointer

NULL Pointer is a pointer which is pointing to nothing. In case, if we don't have address to be assigned to a pointer, then we can simply use NULL.

```
#include <stdio.h>
int main()
{
   // Null Pointer
   int *ptr = NULL;

   printf("The value of ptr is %p", ptr);
   return 0;
}
```

Wild pointer

A pointer which has not been initialized to anything (not even NULL) is known as wild pointer. The pointer may be initialized to a non-NULL garbage value that may not be a valid address.

```
int main()
{
   int *p;  /* wild pointer */

   int x = 10;

   // p is not a wild pointer now
   p = &x;

   return 0; }
```

Q6) Virtual Function

A **virtual** function is a function in a base class that is declared using the keyword **virtual**. Defining in a base class a virtual function, with another version in a derived class, signals to the compiler that we don't want static linkage for this function.

What we do want is the selection of the function to be called at any given point in the program to be based on the kind of object for which it is called. This sort of operation is referred to as **dynamic linkage, or late binding.**

Pure Virtual Functions

It is possible that you want to include a virtual function in a base class so that it may be redefined in a derived class to suit the objects of that class, but that there is no meaningful definition you could give for the function in the base class.

We can change the virtual function area() in the base class to the following -

```
class Shape {
  protected:
    int width, height;

  public:
    Shape(int a = 0, int b = 0) {
      width = a;
      height = b;
    }

    // pure virtual function
    virtual int area() = 0;
};
```

The = 0 tells the compiler that the function has no body and above virtual function will be called **pure virtual function.**

Section C

Q7) a)

Templates are powerful features of C++ which allows you to write generic programs. In simple terms, you can create a single function or a class to work with different data types using templates. Templates are often used in larger codebase for the purpose of code reusability and flexibility of the programs.

```
#include<iostream.h>

#include<conio.h>

template<class t>

void swap(t &x, t &y) {

  t temp = x;

  x = y;

  y = temp;}

void fun(int a, int b, float c, float d) {

  cout << "\na and b before swaping :" << a << "\t" << b;

  swap(a, b);

  cout << "\na and b after swaping :" << a << "\t" << b;

  cout << "\n\nc and d before swaping :" << c << "\t" << d;

  swap(c, d);
```

```
    cout << "\nc and d after swaping  :" << c << "\t" << d;
}
void main() {
    int a, b;
    float c, d;
    clrscr();
    cout << "Enter A,B values(integer):";
    cin >> a>>b;
    cout << "Enter C,D values(float):";
    cin >> c>>d;
    fun(a, b, c, d);
    getch();
}
```

Class Template

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

using namespace std;

template<class T>
// Template Class
class TClassMax {
    T x, y;
  public:
    TClassMax() {
    }
    TClassMax(T first, T second) {
      x = first;
      y = second;
    }
    T getMaximun() {
```

```
    if (x > y)
        return x;
    else
        return y;
  }};
int main() {
  TClassMax <int> iMax; // (100, 75);
  int a, b, i;
  TClassMax <float> fMax; // (90.78, 750.98);
  float c, d, j;
  cout << "Class Template Programs : Generic Programming : Get Maximum Number
\n";
  cout << "Enter A,B values(integer):";
  cin >> a>>b;
  iMax = TClassMax<int>(a, b);
  i = iMax.getMaximun();
  cout << "Result Max Int : " << i;

  cout << "\n\nEnter C,D values(float):";
  cin >> c>>d;
  fMax = TClassMax<float>(c, d);
  j = fMax.getMaximun();
  cout << "Result Max Float : " << j;

  getch();
  return 0;
}
```

b)
```
#include <bits/stdc++.h>
using namespace std;

// driver code
int main()
{
  fstream file;
```

```
// Input stream class to
// operate on files.
ifstream ifile("file.txt", ios::in);

// Output stream class to
// operate on files.
ofstream ofile("file2.txt", ios::out | ios::app);

// check if file exists
if (!ifile.is_open()) {

    // file not found (i.e, not opened).
    // Print an error message.
    cout << "file not found";
}
else {
    // then add more lines to
    // the file if need be
    ofile << ifile.rdbuf();
}
string word;

// opening file
file.open("file2.txt");

// extracting words form the file
while (file >> word) {

    // displaying content of
    // destination file
    cout << word << " ";
}

return 0;
}
```

Q8)a)

```
using namespace std;

class A
{
        int a,b;
        public:
                A(){};
                A(int i,int j)
                {
                        a=i;
                        b=j;

                }
```

```
        void show()
        {
                cout<<a<<"+i"<<b;
        }
        A operator +(A);
};
A A::operator +(A obj)
{
        A temp;
        temp.a=a+obj.a;
        temp.b=b+obj.b;
        return(temp);
}
int main()
{
        A c1(5,6),c2(7,8),c3;
        cout<<"The 1st no. is:";
        c1.show();
        cout<<"\nThe 2nd no. is:";
        c2.show();
        c3=c1+c2;
        cout<<"\nSum is:";
        c3.show();
}
```

```
#include <iostream>
using namespace std;
int main()
{
 double length=0.0,   // The rectangle's length
     width=0.0,     // The rectangle's width
     area=0.0;      // The rectangle's area

 // Get the rectangle's length and rectangle's width..
 getInput(length, width);


 // Get the rectangle's area.
 area = getArea(length, width);

 // Display the rectangle's length, width, and area.
 displayData(length, width, area);

 return 0;
```

)a)

heritance in C++

The capability of a class to derive properties and characteristics from another class is called **Inheritance.** Inheritance is one of the most important feature of Object OrientedProgramming.

Sub Class: The class that inherits properties from another class is called Sub class or Derived Class.

Super Class: The class whose properties are inherited by sub class is called Base Class or Super class.

In C++, we have 5 different types of Inheritance. Namely,

1. Single Inheritance
2. Multiple Inheritance
3. Hierarchical Inheritance
4. Multilevel Inheritance
5. Hybrid Inheritance (also known as Virtual Inheritance)

b) In C++, a derived class object can be assigned to a base class object, but the other way is not possible.

```
class Base { int x, y; };

class Derived : public Base { int z, w; };

int main()
{
   Derived d;
   Base b = d; // Object Slicing,  z and w of d are sliced off
}
```

Object slicing happens when a derived class object is assigned to a base class object, additional attributes of a derived class object are sliced off to form the base class object.