Q.2 a. Explain in detail various OOPs concept in C++. (6)

Answer:

Object oriented programming is method of programming where a system is considered as a collection of objects that interact together to accomplish certain tasks. Objects are entities that encapsulate data and procedures that operate on the data.

In OOPS first a concept known as "Object oriented analysis(OOA)" is used to specify the objects in term of real world requirements, their behaviour and interactions required. The next concept would be the "Object oriented design(OOD)" that converts these realtime requirements as a hierarchy of objects in terms of software development requirement. Finally OOPS is used to implement the requirements using the C++ programming language.

The main purpose of object oriented programming is to simplify the design, programming and most importantly debugging a program. So to modify a particular data, it is easy to identify which function to use. To add additional features it is easy to identify where to add functions and its related data.

b. Write a program in C++ to find the factorial of any number entered through the keyboard. (4)

Answer:

```
include<iostream.h>

        int main()
    {
        int i,f=1,num;
        cout << "Enter the number: ";
        cin >> num;
           for(i=1;i<=num;i++)
           {
             f = f * i;
           }
        cout << "Factorial is = " << f;
     }
```

c. Write a program in C++ to convert decimal to binary. (6)

Answer:

C++ Program for Converting Decimals to Binary
C++ Program to convert decimal number into binary

```
#include <iostream>
using namespace std;
int main()
{
  long dec,rem,i=1,sum=0;
  cout<<"Enter the decimal to be converted:";
  cin>>dec;
  do
  {
    rem=dec%2;
    sum=sum + (i*rem);
    dec=dec/2;
    i=i*10;
  }while(dec>0);
  cout<<"The binary of the given number is:"<<sum<<endl;
  cin.get();
  cin.get();

  return 0;
}
```

Q.3 a. Write a program that accepts number as input and prints a grade depending on the number range, (80 and above) Grade 'A', (60 to 79) Grade 'B', (40 to 59) Grade 'C' and (Below 40) Grade 'F'. You should take the number as input and print the corresponding letter grade. Use *if-then-else* structure OR *switch-case* for your program. (8)

Answer:

b. What is character array? How character array can be initialized? (2+2)

Answer:
Page No. 101

c. Write the rules for using indirection operator (*) and address-of operator (&). (4)

Answer:
Page No. 111

Q.4 a. What do you mean by function overloading? Write a C++ program to swap two integer, float and character variables using function overloading. (2+6)

Answer:

```
#include<iostream.h>
#include<conio.h>
void swap(int &ix,int &iy);
void swap(float &fx,float &fy);
void swap(char &cx,char &cy);
void main()
{
                    int ix,iy;
float fx,fy;
char cx,cy;
clrscr();
cout<<"Enter 2 integers:";
cin>>ix>>iy;
cout<<"Enter 2 floating point no:s:";
cin>>fx>>fy;
cout<<"Enter 2 characters:";
cin>>cx>>cy;
cout<<"\nIntegers:";
cout<<"\nix="<<ix<<"\niy="<<iy;
swap(ix,iy);
cout<<"\nAfter swapping";
cout<<"\nix="<<ix<<"\niy="<<iy;
cout<<"\nFloating point no:s";
cout<<"\nfx="<<fx<<"\nfy="<<fy;
swap(fx,fy);
cout<<"\nAfter swapping";
cout<<"\nfx="<<fx<<"\nfy="<<fy;
cout<<"\nCharacters";
cout<<"\ncx="<<cx<<"\ncy="<<cy;
swap(cx,cy);
cout<<"\nAfter swapping";
cout<<"\ncx="<<cx<<"\ncy="<<cy;
getch();
}
void swap(int &a,int &b)
{
int temp;
temp=a;
a=b;
b=temp;
}
void swap(float &a, float &b)
                    {
float temp;
temp=a;
a=b;
b=temp;
}
void swap(char &a, char &b)
{
char temp;
temp=a;
a=b;
b=temp;
}
```

b. What do you mean by static variable and static function? Give an example. (8)

Answer:

Static class member variables are used commonly by the entire class. It stores values. No different copy of a static variable is made for each object. It is shared by all the objects. It is just like the C static variables.

- It has the following characteristics:
- On the creation of the first object of the class a static variable is always initialized by zero.
- All the objects share the single copy of a static variable.
- The scope is only within the class but its lifetime is throughout the program.

A static function is just like a static variable. a static function has the following properties:

- A function can access only the static variable where both belong to the same class.
- A static function is called using the class name directly and not the object like

class-name :: function-name;

The example given below demonstrates the use of static variable 'count' and static function 'display()':

```
#include<iostream.h>
using namespace std;
class demo
{
int num;
static int count;
public:
void set(void)
{
num=++count;
}
void show(void)
{
cout<<"Number : "<< num<<"\n";
}
static void display(void)
{
cout<<"count : "<<count<<"\n";
}
};
int demo::count;
int main()
{
demo d1,d2;
d1.set();
d2.set();
demo::display();//accessing static function
demo d3;
d3.set();
d1.show();
d2.show();
d3.show();
return 0;
}
```

Q.5 a. What are the similarities and differences between a class and a structure? (8)

Answer:

Structures and classes are the two most important data structures that are used by programmers to build modular programs by using OOP languages, such as Visual Basic .NET, and Visual C#. The following are some of the similarities between a class and a structure:
Access specifiers, such as public, private, and protected, are identically used in structures and classes to restrict the access of their data and methods outside their body.
The access level for class members and struct members, including nested classes and structs, is private by default. Private nested types are not accessible from outside the containing type.
Both can have constructors, methods, properties, fields, constants, enumerations, events, and event handlers.
Both structures and classes can implement interfaces to use multiple-inheritance in code.
Both structures and classes can have constructors with parameter.
Both structures and classes can have delegates and events.

b. What is the relationship between a class and an object? (4)

Answer:

A class acts as a blue-print that defines the properties, states, and behaviors that are common to a number of objects. An object is an instance of the class. For example, you have a class called Vehicle and Car is the object of that class. You can create any number of objects for the class named Vehicle, such as Van, Truck, and Auto.

The new operator is used to create an object of a class. When an object of a class is instantiated, the system allocates memory for every data member that is present in the class.

c. What is the difference between constructor and destructor? (4)

Answer:

- A constructor is called when you want to create a new instance of a class. A destructor is called when you want to free up the memory of an object (when you delete it).
- A constructor constructs the value of an object . A destructor destructs the value created by the constructor for the object.

- Another differentiation is their syntax's :

constructor :

```
    <Class_name>(Arguments)
```

```
        {

                //body of constructor.

        }
```

destructor :

```
        ~<class_name>()

        {

}
```

Q.6 a. What is Operator Overloading? Explain its syntax? Name which operators can-not be overloaded? (8)

Answer:
Page No. 230 – 231, 239

b. Explain with the help of program, how 'new' and 'delete' operators can be overloaded. (8)

Answer:
Page No. 253

Q.7 a. Discuss multiple inheritance with proper syntax and rules in C++. (8)

Answer:
C++ provides us with a very advantageous feature of multiple inheritance in which a derived class can inherit the properties of more than one base class. The syntax for a derived class having multiple base classes is as follows:

```
Class D: public visibility base1, public visibility base2
{
Body of D;
}
```

Visibility may be either 'public' or 'private'.
In case of multiple inheritances, the base classes are constructed in the order in which they appear in the declaration of the derived class. However the constructors for virtual base classes are invoked before any non-virtual base classes. If there are multiple virtual base classes, they are invoked in the order in which they are declared. An example program illustrates the statement.

```
#include<iostream.h>
using namespace std;
class M
{
protected:
int m;
public:
```

```
M(int x)
{
m=x;
cout<< "M initialized\n";
}
};
class N
{
protected:
int n;
public:
N(int y)
{

n=y;
cout<< "N initialized\n";
}
};
class P: public N, public M
{ int p,r;
public:
P(int a,int b,int c, int d):M(a),N(b)
{
p=c;
r=d;
cout<< "P initialized\n";
}
};
void main()
{
P p(10,20,30,40);
}
```

b. Explain, with the help of a program, the order in which constructors are invoked when there is multiple inheritance. (4)

Answer:

The order in which constructor is invoked when there is multiple inheritance is order in which the class is derived that is constructor of base class will invoke first and then the constructor of derived class and so on. The destructor will invoke in reverse order than constructor. That is destructor of derived class will invoke first and then base class. Example

```
Class a
{
a()
{
cout<<"constructor a";
}
~a()
{
```

```
cout<<"destructor a";
}
};
class b : public a
{
b()
{
cout<<"constructor b";
}
~b()
{
cout<<"destructor b";
}
};
void main()
{
b b1;
}
```

c. What is virtual function? Explain the syntax of pure virtual function.(2+2)

Answer:

Q.8 a. What is an exception? Explain the usage of nested try-catch block. Is it necessary that number of catch blocks should be equal to the number of try blocks? Justify. (2+3+3)

Answer:

b. Define a class template Queue with put() and get() operations. Using this create a Queue of integers in main() and add two elements to the queue. (8)

Answer:

```
#include<iostream.h>
const int max=3;
template<class type>
class queue
{
type qu[max];
int head,tail,count;
public:
class full
{
};
class empty
{
};
queue()
{
head=-1;
tail=-1;
count=0;
```

```
}
void put(type var)
{
if(count>=max)
throw full();
qu[++tail]=var;
++count;
if(tail>=max-1)
tail=-1;
}

type get()
{
if(count<=0)
throw empty();
type temp=qu[++head];
--count;
if(head>=max-1)
head=-1;
return temp;
}
};
int main()
{
queue<float> q1;
float data;
char choice='p';
do
{
try
{
cout<<"\enter 'x' to exit, 'p'for put,'g'for get:";
cin>>choice;
if(choice=='p')
{
cout<<"Enter data value:";
cin>>data;
q1.put(data);
}
if(choice=='g')
cout<<"Data="<<q1.get()<<endl;
}
catch(queue<float>::full)
{
cout<<"Error:queue is full."<<endl;
}
catch(queue<float>::empty)
```

```
{
cout<<"Error: queue is empty"<<endl;
}
}while(choice!='x');
return 0;
}
```

Q.9 a. Give the difference between manipulators and ios member functions. (8)

Answer:

The basic difference between manipulators and ios member functions in implementation is as follows:

The ios member function returns the previous format state which can be of use later on but the manipulator does not return the previous format state. Whenever we need to save the old format states, we must use the ios member functions rather then the manipulators.

Example: an example program illustrating the formatting of the output values using both manipulators and ios functions is given below:

```
#include<iostream.h>
#include<iomanip.h>
using namespace std;
int main()
{
cout.setf(ios::showpoint);
cout<<setw(5)<<"\n"
<<setw(15)<<"Inverse_of_n"
<<setw(15)<<"Sum_of_terms\n\n";
double term,sum=0;
for(int n=1;n<=10;n++)
{
term=1.0/float(n);
sum+=term;
cout<<setw(5)<<n
<<setw(14)<<setprecision(4)
<<setiosflags(ios::scientific)<<term
<<setw(13)<<resetiosflags(ios::scientific)
<<sum<<endl;
}
return 0;
}
```

b. Explain the meaning of following state flags:

(i) bad() (ii) fail()

(iii) eof() (iv) good() (2×4)

Answer:

Page No. 366

TEXT BOOK

I. C++ and Object-Oriented Programming Paradigm, Debasish Jana, Second Edition, PHI, 2005 (TB-I:)