Chapter 1
Principles of Object Oriented Programming (14 Marks)

Q1. Give Characteristics of object oriented programming?

Or

Give features of object oriented programming?

Ans:

1. Emphasis (focus) is on data rather than procedure.
2. Programs are divided into what are known as objects.
3. Data is hidden and can not be access by external functions.
4. Objects may communicate with each other through functions.
5. New data and functions can easily be added when ever necessary.
6. Follows bottom- up approach in programming design.

Q2. What do you meant by object oriented programming language? Enlist any six object oriented programming languages.

Ans:

Object-oriented programming (**OOP**) is a programming paradigm that uses "objects" to design applications and computer programs. **Object-oriented programming** include features such as data abstraction, encapsulation, modularity, polymorphism, and inheritance.

Object oriented programming languages are as follows:

1. C++
2. Java
3. Simula 67
4. Object Pascal
5. Smalltalk
6. C#.NET
7. Objective C

Q3. List Applications of OOP (Object-oriented programming).

Ans:

1. Real time System
2. Simulation and Modelling
3. Hypertext And Hypermedia
4. Decision support system
5. CAM/CAE/CAD System
6. Office Automation System
7. Artificial intelligence and expert system

Q4. Explain the term object?

Or

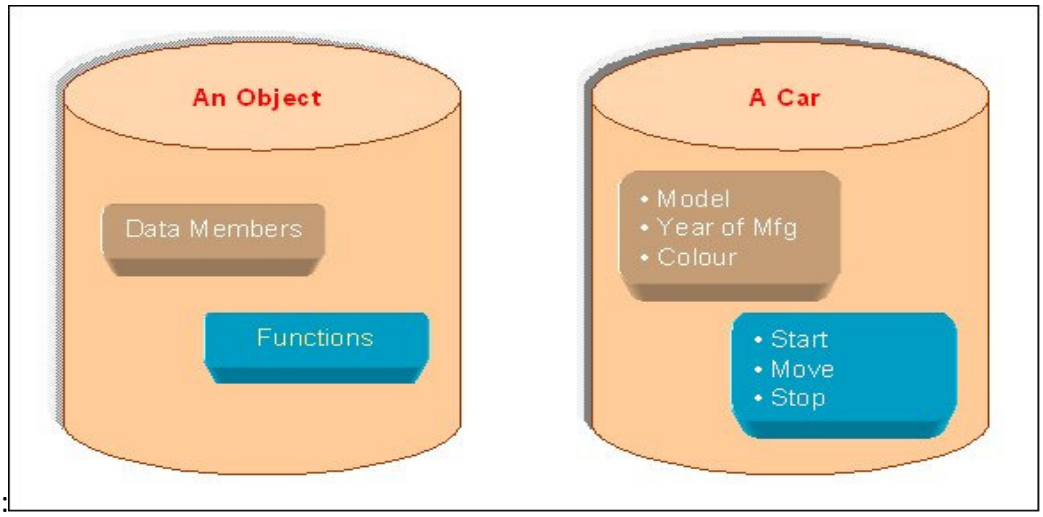
What is object? Explain with example?

Or

What are objects? How they are created?

Ans:

1. Objects are the basic runtime entities in an object oriented system.
2. Object is the basic unit of object-oriented programming.
3. Objects may represent a person, a place, Bank account, a table of data, or any item that the program has to handle.
4. Objects are the class variables.
5. Example1:



Example 2: Mango, Apple, Grapes are the objects of class ***fruit***.

Syntax to create an object is.

Fruit mango, apple;

Where ***fruit*** is class name. This will create an object ***mango*** and ***apple*** of type ***fruit***.

Example to create an object:

```
class fruit
{
        ……
        ……
} mango, apple;
```

This will create an object ***mango*** and ***apple*** of type ***fruit***.

Q5. Define class with syntax?

Or

Define class. Give syntax of class declaration?

Or

What is class? Give examples?

Ans:

A collection of objects with similar data type is known as class.

Classes are the user-defined data types and behave like built-in types in programming. A class is a way to bind the data and its associated function together.
Example : Mango, Apple, Grapes are the objects of class ***fruit***.
Syntax for class declaration.

```
class class_name
{
        private:
                variable declaration;
                function declaration;
        public:
                 variable declaration;
                function declaration;
};
```

Example:

```
class student
{
        int roll_no;                    // variable declaration
        char name[20];
public:
        void getdata();                 // Function declaration
        void putdata();
}; // ends with semicolon
```

Q 6 Explain syntax of switch case statement with example.

A switch statement allows a variable to be tested for equality against a list of values. Each value is called a case, and the variable being switched on is checked for each case.

Syntax

```
switch(expression)
{
  case constant-expression  :
    statement(s);
    break;
  case constant-expression  :
    statement(s);
    break;


  default :
    statement(s);
}
```

For example,

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

int main ()
{
  char grade = 'D';
  switch(grade)
  {
    case 'A' :
      cout << "Excellent!" << endl;
      break;
    case 'B' :
            cout << "Very Good!" << endl;
      break;
    case 'C' :
      cout << "Good" << endl;
      break;
    case 'D' :
      cout << "You passed" << endl;
      break;
    case 'F' :
      cout << "Better try again" << endl;
      break;
    default :
      cout << "Invalid grade" << endl;
  }
  cout << "Your grade is " << grade << endl;

  getch();
  return 0;
}
```

Output:

```
You passed
Your grade is D
```

Q7. Give syntax and example of defining structure and declaring structure variable.

Ans:

Structure is collection elements with dissimilar data type.

Syntax:

```
Struct structure_name
{
        Structure element 1;
Structure element 2;
Structure element 3;
};
```

Example:

```
struct book
{
        char book_name;
        float price;
        int pages;
} b1, b2, b3;
```

Above example defines a new data type called struct book. Each variable of this data type will consist of a character variable called ***book_name***, a float variable called ***price*** and an integer variable called ***pages***; and the ***b1, b2, b3*** are the variables of type ***struct book***.

Q8. Explain difference between structure and class with example.

Ans:

Structure	**Class**
1. Structure is collection of dissimilar data types.	1. Class is collection of objects.
2. Structure members are by default public.	2. Class members are by default private.
3. Structure does not support inheritance and data hiding.	3. Class supports inheritance and data hiding.
4. Structure does not support member function declaration.	4. Class supports member function declaration.

5. Syntax:
struct structure_name
{
 	
Variabledeclaration;
};
	5. Syntax:
	class class_name
	{
	private: //optional
	variable declaration;
	function declaration;
	public:
	variable declaration;
	function declaration;
	};

6. Example: struct book {	6.Example: class student {

char book_name;
float price;
int pages;
} b1, b2, b3;
	int roll_no; // variable
	declaration
	char name[20];
	public:
	void getdata(); // Function declaration
	void putdata();
	}; // ends with semicolon

Q9. Draw the structure of C++ Program.

Ans:

A typical C++ program would contain four sections.

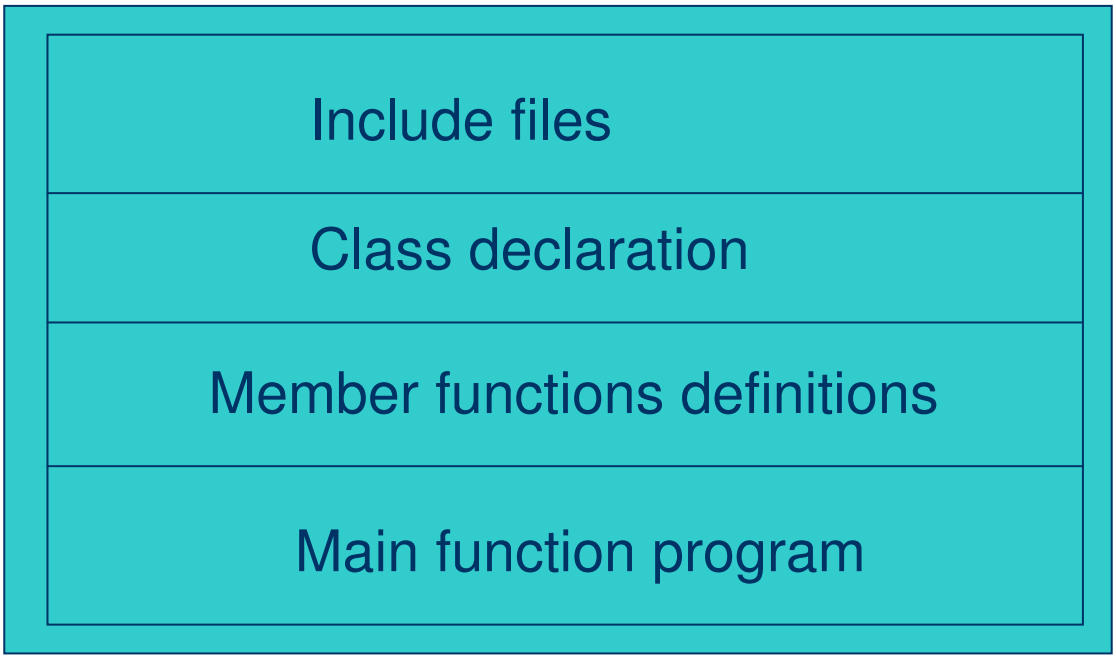


Fig. Structure of C++ Program

It is a common practice to organize a program into three separate files. The class declarations are placed in a header file and the definitions of member functions go into another file.

Example:

```
//program to read employee details and to output the data
#include <iostream.h>                 // Pre-processor directive
class employee                        //Class Declaration
{
  private:
    char empname[50];
    int empno;
  public:
    void getvalue()                           //Function Definition
    {
```

```
    cout<<"INPUT EMP NAME:";
    cint>>empname;
    cout<<"INPUT EMP NO:";
    cint>>empno;
  }
void displayvalue()
  {
    cout<<"EMP NAME:"<<empname;
    coout<<EMP NO:"<<empno;
  }
};

void main()
{
  employee e1;                    //Creation of Object
  e1.getvalue();
  e1.displayvalue();
}
```

Q10.Which are the input and output operator in C++? Give example.

OR

Explain working of insertion and extraction operators in C++ with the help of suitable example.

Ans:

Input Operator:

cin >> number1;

The above statement is an input statement and causes the program to wait for the user to type in a number. The identifier ***cin*** is a predefined object in C++ that corresponds to the standard input stream. Here this, this stream represent keyboard.

The operator ***>>*** is known as ***extraction*** or ***get from*** operator. It takes value from keyboard and assigns it to the variable on its right. It is similar to ***scanf()*** operation in C programming.

Output Operator:

cout << number1;

The above statement is an output statement and causes the program print value of ***number1*** Variable on Screen. The identifier ***cout*** is a predefined object in C++ that corresponds to the standard output stream. Here this, this stream represent screen.

The operator **<<** is known as ***insertion*** or ***put to*** operator.

It is similar to ***printf()*** operation in C programming.

Q11. Differentiate between Object Oriented Programming and Procedure Oriented Programming.

	Procedure Oriented Programming	Object Oriented Programming
Divided Into	In POP, program is divided into small parts called **functions**.	In OOP, program is divided into parts called **objects**.
Importance	In POP, Importance is not given to **data** but to functions as well as **sequence** of actions to be done.	In OOP, Importance is given to the data rather than procedures or functions because it works as a **real world**.
Approach	POP follows **Top Down approach**.	OOP follows **Bottom Up approach**.
Access Specifiers	POP does not have any access specifier.	OOP has access specifiers named Public, Private, Protected, etc.
Data Moving	In POP, Data can move freely from function to function in the system.	In OOP, objects can move and communicate with each other through member functions.
Expansion	To add new data and function in POP is not so easy.	OOP provides an easy way to add new data and function.
Data Hiding	POP does not have any proper way for hiding data so it is **less secure**.	OOP provides Data Hiding so provides **more security**.
Overloading	In POP, Overloading is not possible.	In OOP, overloading is possible in the form of Function Overloading and Operator Overloading.
Examples	Examples of POP are: C, VB, FORTRAN, Pascal.	Examples of OOP are: C++, JAVA, VB.NET, C#.NET.

Q 12. What is data abstraction?

Abstraction refers to showing only the essential features of the application and hiding the details. In C++, classes can provide methods to the outside world to access & use the data variables, keeping the variables hidden from direct access.

For example, When you send an sms you just type the message, select the contact and click send, the phone shows you that the message has been sent, what actually happens in background when you click send is hidden from you as it is not relevant to you.

Q 13. What is data encapsulation?

Binding of is data members and member functions together in class is called data encapsulation. Encapsulation also leads to data abstraction or hiding. In other words, **Encapsulation** is defined as wrapping up of data and information under a single unit.

Q14. Differentiate between C and C++

Difference between C and C++		
Sr. No	**C**	**C++**
1	When compared to C++, C is a subset of C++.	C++ is a superset of C. C++ can run most of C code while C cannot run C++ code.
2	C supports procedural programming paradigm for code development.	C++ supports both procedural and object oriented programming paradigms.
3	C does not support for polymorphism, encapsulation, and inheritance.	C++ supports polymorphism, encapsulation, and inheritance.
4	In C, data and functions are separate and free entities.	In C++, data and functions are encapsulated together in form of an object.
5	C does not support information hiding.	C++ not supports information hiding by encapsulation.
6	C, is a function driven language.	While, C++ is an object driven language.
7	C does not support function and operator overloading.	C++ supports both function and operator overloading.
8	C uses functions for input/output. For example scanf and printf.	C++ uses objects for input output. For example cin and cout.
9	C has no support for virtual and friend functions.	C++ supports virtual and friend functions.

Q 15 Explain with suitable example, syntax of for loop in C++.

For loop is a looping statement that allows us to execute a statement or group of statements multiple times.

Syntax of for loop

```
for(initialization; condition ; increment/decrement)
{
C++ statement(s);
}
```

e.g.

```
#include <iostream.h>
#include<conio.h>
int main()
{
        int arr[]={21,9,56,99, 202};
```

```
        for(int i=0; i<5; i++)
        {
                cout<<arr[i]<<endl;
        }
        getch();
        return 0;
}
```

Output:

21
9
56
99
202

Q 16 State the use of scope resolution operator and its use in C++.

OR

what is the significance of scope resolution operator?

OR

What is use of scope resolution operator.

Ans

In C++, scope resolution operator is **::**. It is used for following purposes.

1) **To access a global variable when there is a local variable with same name.**
2) To define a function outside a class.
3) To access a class's static variables.
4) In case of multiple Inheritance.

e.g.

```
#include<iostream.h>
int x;  // Global x
int main()
{
  int x = 10; // Local x
  cout << "Value of global x is " << ::x;
  cout << "\nValue of local x is " << x;
  return 0;
}
```

Output :

Value of global x is 0
Value of local x is 10

Q 17 State use of new operator.

OR

List two memory management operators in C++ and state its use in one line.
OR
Explain memory management operators in C++ with suitable example.
Ans
new operator
The **new operator** allocates memory dynamically to a variable and returns the base address of that memory to a pointer.

Syntax
pointer_name = new datatype;

Delete operator
Delete operator is used to release the memory allocated in the heap using new operator.

Syntax

delete pointer_name;

Example,

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

void main()
{

        int size,i;
        int *ptr;
        cout<<"\n\tEnter size of Array : ";
        cin>>size;

        ptr = new int[size];

        for(i=0;i<5;i++)
        {
                cout<<"\nEnter any number : ";
                cin>>ptr[i];
        }

        for(i=0;i<5;i++)
        {
                cout<<ptr[i]<<", ";
```

```
        }

        delete[] ptr;
        getch();
        return 0;

}
```

Output :

```
Enter size of Array : 5

Enter any number : 78
Enter any number : 45
Enter any number : 12
Enter any number : 89
Enter any number : 56

78, 45, 12, 89, 56,
```

Q 18 State the data types available in C++ with their storage sizes.

Type	Keyword	Typical Bit Width
Character	char	1byte
Integer	int	2bytes
Floating point	float	4bytes
Double floating point	double	8bytes

Q 19 Differentiate between do..while and while loops on the basis of syntax.

Sr. No.	do…while	while
1	It is Exit controlled loop	It is Entry controlled loop
2	Executes at least once	Executes only if condition is satisfied
3	Syntax: do	Syntax: while(condition)

	{
	//statements in loop
	}while(condition);
		{
		//statements in loop
		}

4.	The block control condition is available at the starting point of the loop	The block control condition is available at the end point of the loop

Q 20 Explain syntax of any two String functions.

Ans

String is a collection of characters.

1) **String Copy**

Snytax : strcpy(to_string, from_string)

The strcpy() function copies the string pointed by *from_string* (including the null character) to the character array *to_string*. This function returns character array *to_string*.

2) **String Length**

Syntac: strlen(string)

strlen(str) returns the length of the string pointed to by str, i.e., the number of characters excluding the null terminator.

3) **Concatenation of Strings**

Syntax: strcat(string_1, string_2)

The strcat() function appends s2 to the end of s1. String s2 is unchanged.

4) **Comparison of Strings**

Syntax: strcmp(string_1, string_2)

The strcmp(str_1, str_2) function compares two character by character. If the first character of two strings is equal, next character of two strings is compared. This continues until the corresponding characters of two strings are different or a null character '\0' is reached.
Return Value from strcmp()

Return Value Remarks

0- if both strings are identical (equal)
Negative- if the ASCII value of first unmatched character is less than second.
Positive integer- if the ASCII value of first unmatched character is greater than second.
e.g.

```
#include <iostream.h>
#include<string.h>
int main()
{
char s1[21],
s2[11];
strcpy(s1, “hello”);
```

```
strcpy(s2, " there");
strcat(s1, s2);
cout << s1 << endl;
cout << s2 << endl;
cout<< "length of concatenated string is: "<<strlen(s1);
if(strcmp(s1,s2))
{
cout << "Strings are different.\n";
}
Else
{
cout << "Strings are equal.\n";
}
return(0);
}
```

Q 21 Write a C++ program that displays first 10 odd numbers.

```
#include <iostream.h>
int main()
{
   int i=1;
   while(i != 20)
   {
         if(i%2!=0)
                  cout << "Odd Number = " << i;
         i++;
   }
   return 0;
}
```

OR

```
#include <iostream.h>
int main()
{
   int i=1;
   while(i != 20)
   {
         cout << "Odd Number = " << i;
         i= i+2;
   }
   return 0;
}
```

Q 22 Define structure. Give syntax of defining structure.
OR
Define structure with its syntax.
Structure is a compound data type that contains different variables of different types. For example, you want to store Student details like student name, student roll num, student age.
Syntax:

```
struct struct_name
{
        //variables declaration;
};
```

e.g.

```
struct Student
{
   char stuName[30];
   int stuRollNo;
   int stuAge;
};
```

Q 23 write a C++ program to define a structure 'Student' having variables name, rollno, age. Accept and display data for one structure object.

```
#include <iostream>
struct Student
{
  char stuName[30];
  int stuRollNo;
  int stuAge;
};
int main()
{
  Student s;
  cout<<"Enter Student Name: ";
  cin.getline(s.stuName, 30);
  cout<<"ENter Student Roll No: ";
  cin>>s.stuRollNo;
  cout<<"Enter Student Age: ";
  cin>>s.stuAge;
  cout<<"Student Record:"<<endl;
  cout<<"Name: "<<s.stuName<<endl;
  cout<<"Roll No: "<<s.stuRollNo<<endl;
  cout<<"Age: "<<s.stuAge;
  return 0;
```

```
}
```

Output:

```
Enter Student Name: Aksha
ENter Student Roll No: 006
Enter Student Age: 8
Student Record:
Name: Aksha
Roll No: 006
Age: 8
```

Q 24 What is typecasting? Give example.

OR

Explain typecasting with suitable example.

Typecasting is making a variable of one type to act like another type for one single operation. To typecast something, simply put the type of variable you want the actual variable to act as inside parentheses in front of the actual variable.

```
#include <iostream.h>
int main()
{
        for ( int x = 0; x < 128; x++ )
        {
                cout<< x <<". "<< (char)x <<" ";
        }
        return 0;
}
```

Q 25 Explain logical operators with example.

Logical Operators are used with binary variables. They are mainly used in conditional statements and loops for evaluating a condition.

Logical operators in C++ are: &&, ||, !

Consider that A = 0 and B = 0

Operator	Description	Example
Logical AND (&&)	If both the operands are non-zero then only condition becomes true	(A && B) is false.
Logical OR (||)	If both the operands are zero then only condition becomes false	(A || B) is true.

Logical NOT (!)	It will reverses the state of its operand i.e true will become false	(!A) is true.

Q 26 Explain relational operators with example.

C++ have six relational operators in C++: ==, !=, >, <, >=, <=

Consider that A = 40 and B = 20

Symbol	Meaning	Example
>	Greater than	A > B returns true
<	Less than	A < B returns false
>=	Greater than equal to	A >= B returns false
<=	Less than equal to	A <= B returns false
==	Equal to	A == B returns false
!=	Not equal to	A != B returns true

Q 27 Explain bitwise operators with example.

There are six bitwise Operators: &, |, ^, ~, <<, >>

Assume if A = 60; and B = 13; now in binary format they will be as follows −

A = 0011 1100

B = 0000 1101

Operator	Description	Example
&	Binary AND Operator copies a bit to the result if it exists in both operands.	(A & B) will give 12 which is 0000 1100
|	Binary OR Operator copies a bit if it exists in either operand.	(A | B) will give 61 which is 0011 1101
^	Binary XOR Operator copies the bit if it is set in one operand but not both.	(A ^ B) will give 49 which is 0011 0001
~	Binary Ones Complement Operator is unary and has the effect of 'flipping' bits.	(~A) will give -61 which is 1100 0011 in 2's complement form due to a signed binary number.
<<	Binary Left Shift Operator. The left operands value is moved left by the number of bits	A << 2 will give 240 which is

	specified by the right operand.	1111 0000
>>	Binary Right Shift Operator. The left operands value is moved right by the number of bits specified by the right operand.	A >> 2 will give 15 which is 0000 1111

Q 28 What is conditional operator? Give example.
OR
Explain ternary operator with example.
This operator evaluates a boolean expression and assign the value based on the result.
Syntax:

variable num1 = (expression) ? value if true : value if false

If the expression results true then the first value before the colon (:) is assigned to the variable num1 else the second value is assigned to the num1.

e.g.

```
#include <iostream.h>

int main()
{
  int num1, num2; num1 = 99;
  num2 = (num1 == 10) ? 100: 200;
  cout<<"num2: "<<num2<<endl;
  num2 = (num1 == 99) ? 100: 200;
  cout<<"num2: "<<num2;
  return 0;
}
```

Output:

```
num2: 200
num2: 100
```

Q 29 Explain sizeof() operator with suitable example.
OR
Give significance of sizeof() operator.

Ans

• sizeof is a compile-time operator used to calculate the size of data type or variable.
• sizeof operator will return the size in integer format.

Syntax of sizeof Operator:

sizeof(data type)

Data type include variables, constants, classes, structures, unions, or any other user defined data type.

Example:

```
#include <iostream.h>
int main()
{
  int i;
  char c;
  cout << "Size of variable i : " << sizeof(i) << endl;
  cout << "Size of variable c : " << sizeof(c) << endl;
  return 0;
}
```

Output:

Size of variable i : 4
Size of variable c : 1

Q 30 Explain if statement with syntax and example.

An if statement consists of a boolean expression followed by one or more statements.

Syntax

```
if(boolean_expression)
{
  // statement(s) will execute if the boolean expression is true
}
```

If the boolean expression evaluates to true, then the block of code inside the if statement will be executed. If boolean expression evaluates to false, then the set of code after the closing curly brace will be executed.

For example,

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

int main ()
{
  int a = 10;
```

```
  if( a < 20 )
  {
    cout << "a is less than 20;" << endl;
   }
  cout << "value of a is : " << a << endl;

  getch();
  return 0;
}
```

Output:

```
a is less than 20;
value of a is : 10
```

Q 31 Explain if-else statement with syntax and example.

An if statement can be followed by an optional else statement, which executes when the boolean expression is false.

Syntax

```
if(boolean_expression)
{
        // statement(s) will execute if the boolean expression is true
}
else
{
        // statement(s) will execute if the boolean expression is false
}
```

If the boolean expression evaluates to true, then the if block of code will be executed, otherwise else block of code will be executed.

For example,

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

int main ()
{
  int a = 100;
  if( a < 20 )
```

```
  {
    cout << "a is less than 20;" << endl;
  }
  else
  {
    cout << "a is not less than 20;" << endl;
  }
  cout << "value of a is : " << a << endl;

  getch();
  return 0;
}
```

Output:

```
a is not less than 20;
value of a is : 100
```