Programming in C++

Basic Concepts of C++

- **Keywords: Keywords are the certain reserved words that convey a special meaning to thecompiler. These are reserve for special purpose and must not be used as identifier name.**
  Eg:- for , if,else , this , do, etc.

- **Identifiers: Identifiers are programmer defined names given to the various program elements such as variables, functions, arrays, objects, classes, etc.. It may contain digits, letters andunderscore, and must begin with a letter or underscore.**

C++ is case sensitive as it treats upper andlower case letters differently. A keyword cannot be used as an identifiers.

The following are somevalid identifiers:

Pen, time580, s2e2r3, _dos, _HJI3, _JK

- **Data Types in C++: Data types are means to identify the types of data and associated operations of handling it. Data types in C++ are of three types:**

1. Fundamental or Built-in data types: These data types are already known to compiler. These are the data types those are not composed of other data types. There are following fundamental datatypes in C++:

(i) int data type (for integer) (ii) char data type (for characters)
(iii) float data type (for floating point numbers) (iv) double data type

2. Derived data types:- They are derived from fundamental data types. These are:-

i) Array ii) Pointer iii) Function iv) Reference

3. User-defined data types :- which are defined by the User.

i) Structure ii) Class iii) Union iv) Enumerated

- **Data Type Modifiers: There are following four data type modifiers in C++ , which may be used to modify the fundamental data types to fit various situations more precisely:**

i) signed ii) unsigned iii) long iv) short

- **Variables: A named memory location, whose contains can be changed with in program execution is known as variable. OR**

A variable is an identifier that denotes a storage location, which contains can be varied during program execution.

Declaration of Variables: Syntax for variable declaration is:

datatypes variable_name1, variable_name2, variable_name3,……. ;

We can also initialize a variable at the time of declaration by using following syntax:

datatypesvariable_name = value;

In C++ both the declaration and initialization of a variable can be done simultaneously at the place where the variable is used first time this feature is known as dynamic initialization. e.g.

```
floatavg;
avg = sum/count;
```

then above two statements can be combined in to one as follows:

```
floatavg = sum/count;
```

- **Constant: A named memory location, whose contains cannot be changed with in program execution is known as constant. OR**

A constant is an identifier that denotes a storage location, which contains cannot be varied during program execution.

Syntax for constant declaration is: **constdatatypesconstant_name = value ;**

e.g:-const float pi = 3.14;

- **Conditional operator (? :):-The conditional operator (? :) is a ternary operator i.e., it require three operands. The general form ofconditional operator is:**

expression1? expression2: expression3 ;

Where expression1 is a logical expression, which is either true or false.

If expression1 is true then expression 2 will execute otherwise expression 3 will be executed.

- **Type Conversion: The process of converting one predefined data type into another is called typeconversion. Two forms are:-**

i) Implicit type conversion:- An implicit type conversion is a conversion performed by the compiler without programmer's intervention. An implicit conversion is applied generally wheneverdifferent data types are intermixed in an expression. The C++ compiler converts all operands uptothe data type of the largest data type's operand, which is called type promotion.

(ii) Explicit type conversion :- An explicit type conversion is user-defined that forces an expression to be of specific data type.

Some important Syntax in C++:

1. if Statement :- **Syntax:- if (condition)** **{** **Statement1;** **}** **else** **{** **Statement2;** **}**	**3. Switch Statement:-** **switch (expression/variable)** **{ case value_1: statement-1;** **break;** **case value_2: statement-2;** **break;** **:** **case value_n: statement n;** **break;** **[default: statement m]** **}**
2. if else if ladder:- **Syntax:- if (condition)** **{** **Statement1;** **}** **elseif(condition)** **{** **Statement2;** **}** **else** **{** **Statement3;** **}**	**5. while Loop:** **while (loop_condition)** **{** **Loop_body;** **}**
4. The for Loop: **for(initialization;Condition;update_expression)** **{** **Body of loop;** **}**	**6. do-while loop:** **do** **{** **Loop_body;** **}** **while (loop_condition);**

- **Functions :- Function is a block of statement that are used to perform some specific task.**

1. Built-in Functions (Library Functions) :- The functions, which are already defined in C++ Library (in any header files) and a user can directly use these function without giving their definition is known as library functions. e.g., sqrt( ), toupper( ), isdigit( ), abs() etc.

Following are some important Header files and useful functions within them :
stdio.h (standard I/O function) gets( ) , puts( )
ctype.h (character type function) isalnum( ) , isalpha( ), isdigit ( ), islower (),isupper ( ), tolower (), string.h (string related function) strcpy ( ), strcat ( ), strlen( ), strcmp( ),strcmpi( ) , strrev(),
strupr( ) , strlwr( )
math.h (mathematical function) fabs ( ), pow ( ), sqrt ( ), sin ( ), cos ( ), abs ( )
stdlib.h randomize ( ), random ( )

2. User-defined function :- The functions which are defined by user for a specific purpose is known as user-defined function. For using a user-defined function it is required, first define it and then using.
Declaration of user-defined Function:
Return_typefunction_name(List of formal parameters)
{
Body of the function;
}

- **Calling a Function:- When a function is called then a list of actual parameters is supplied that**

should match with formal parameter list in number, type and order of arguments.
Syntax for calling a function is:
function_name(list of actual parameters);

- **Call by Value (Passing by value) :- The call by value method of passing arguments to a functioncopies the value of actual parameters into the formal parameters , that is, the function creates its owncopy of argument values and then use them, hence any chance made in the parameters in functionwill not reflect on actual parameters . The above given program is an example of call by value.**

- **Call by Reference (Passing by Reference) :- The call by reference method uses a different**

mechanism. In place of passing value to the function being called , a reference to the original variable is passed . This means that in call by reference method, the called function does not createits own copy of original values , rather, its refers to the original values only by different names i.e. reference . Thus the called function works the original data and any changes are reflected to theoriginal values.

- **Formal Parameters:- The parameters that appear in function definition are formal parameters.**

- **Actual Parameters :- The parameters that appears in a function call statement are actual parameters.**

- **Class :- A class is a group of Object that share common properties and relationships.**

Basically a class is a collection of data (data member) and functions (member functions). It can be seen as a blue print for the object. No memory is allocated when aclass is created. Memory is allocated only when an object is created.

- **Object :- An Object is an instance of the class.i.e It is a run time entity.**
- **Data member:- The data declared within the class.**
- **Member functions :- Member functions are the methods which are declared/defined inside the classand operate upon the data member.**
- **Data Abstraction: - Data abstraction refers to the act of representing essential features without knowing its background details.**
- **Data Encapsulation:-The Wrapping up of data and function together in a single unit called class.**
- **Data hiding:- Hides internal object details (data members). Data hiding ensures exclusive dataaccess to class members and protects object integrity by preventing any changes.**

Inheritance: Inheritance is the process of creating a new class from an existing class or base class.

- **Base Class :- The class from which methods and data members are derived to new class is knows asbase class. The base class is also known as parent class or super class.**
- **Derived Class:- The class that is deriving data and methods from base class is called derived class.Derived class is also known as a child class or sub class.**

- **Polymorphism:- Poly means many and morphism meansmore than one form. Refers to the abilityof processing of data in more than one form.**

Access specifier :-private, protected, public (default access specifier is private)

Accessibility of private, protected and public members

Accessibility	Private	Protected	Public
Through member functions	Yes	Yes	Yes
Through object of the class	No	No	Yes
Through derived class	No	Yes	Yes

Syntax of a class:-

```
class class_name
{
private:
 declaration of data member;
 declaration/definitionmember function;
protected:
 declaration of data member;
 declaration/definition member function
public:
 declaration of data member;
 declaration/definition member function
};
```

Eg:-

```
class student
{
private:
 char name[30];
int age;
int marks;
protected:
 char grade;
public:
 void getdata();
 void showdata();
};
```

- **Referencing class members:- All the data members of the class are directly accessible to the**

member function of that class. They don't need any object name to be prefixed before it but fromoutside the class any reference to the data member is done with the dot (.) operator.
syntax for creating an object:
<class_name><Object_name>;
Example:
student s1;

- **Accessing members from object of the class:- A data member and member function declared underpublic access specifier can be assessed by the objects directly.**

Syntax:-objectname.memberfunction;
e.g:- s1.getdata();
s1.showdata();

- **Defining Member functions:- Member functions of the class can be defined in thefollowing two ways:-**

(a) Inside the class definition (inline function):- In this method, the function is defined within the class body and are treated as inline by default.
(b) Outside the class definition:-In this way function prototype is declared within class body and function is defined outside the classwith the help of Scope Resolution operator (::).

Syntax for defining a member functionoutside the class definition.	Example for defining a member functionoutside the class definition.

<returntype><class name> :: <functionname>(parameter list)
{
body of the function;
}
	void student::showdata()
	{
	cout<<"\n Name "<<name;
	cout<<"\n Age "<age;
	cout<"\n Marks"<marks;
	}

- **<u>Constructors and Destructors</u>**

<u>Constructor</u>:- A constructor is a special member function whose task is to initialize the objects of its class. Constructor name is same as a class name. The constructor is invoked whenever an object of the associated class is created.
Special characteristics of Constructors

1. **A constructor name is same as the name of class.**
2. **They are invoked automatically when the object are created.**
3. **It should nothave any return type not even void.**
4. **Constructor should bedeclared in the public section.**
5. **Constructors are used to initialize the data members of the class.**
6. **They cannot be static.**
7. **All object of the class having a constructor are initialized before their use.**
8. **They cannot be declared const or volatile but a constructor can be invoked as a const and volatile object.**
9. **They cannot be inherited.**
10. **We cannot refer to the address of constructor.**

Syntax:-	Example:-

class classname
{
public:
classname(parameter list);
};
	class Abc
	{
	public:
	Abc();
	};

Constructor Defined inside the class definition	Constructor Defined outside the class definition

class Abc
{
int a;
public:
int b;
Abc()
{
a=b= 20;
}
};
	class Abc
	{
	int a;
	public:
	int b;
	Abc();
	};
	abc : : abc()
	{
	a=b= 20;
	}

Types of Constructors

1. Default Constructor (No argument constructor):- A default constructor accepts no parameters. The default constructor initializes the data member by the dummy values. When no constructor is defined in the class, compiler provides or supplies the defaultconstructor.

Eg:-Abc a;

2. Parameterized Constructor:-A constructor with some parameters list is called parameterized constructor.i.e constructors with arguments are known as parameterized constructors.

It allow the user to initialize various data elements of different objects with different values when they are created.

3. Overloaded Constructor:- When we use more than one constructor with different arguments in a class.

4. Copy Constructor:-A constructor that accepts a reference to an instance of its own class as an argument is called as Copy Constructor. A copy constructor is used to create new object with the similar values of existing object. A copy constructor is invoked when one object is defined and initialized with another object of the same class.

OR

A copy constructor is a constructor that can be used to initialize one object with the value of another object of same class during declaration i.eit is a special constructor that can be used to declare and initialize one object from another object.

Syntax for declaration of copy constructor:-
(classname&obj)
for example:- Student(Student &s)

```
Eg:     Hello H1(5,10);      //H1 is intitalize with value.
        Hello H2(H1);        //H2 will copy the contents of H1
        Hello H3= H1;        //H3 will copy the contents of H1
```

Example of three different types of constructors. (default, parameterize, copy).

```
class student
{
introllno;
float percentage;
public:
student() // default constructor
{
rollno=0;
percentage=0.0;
}
student(intrno,float p) //parameterized constructor
{
rollno=rno;
percentage=p;
}
student(student &s) // copyconstructor
{
rollno=s.rollno;
percentage=s.percentage;
 }
void display()
{
cout<<"RNo. "<<rollno;
cout<<"\n per "<<percentage;
}
};
void main()
{
student s;       //call for the default constructor
student s1(5,88.5);     //call for the parametrized constructor
student s2=s1;//call for the copy constructor
s.display();
s1.display();
s2.display();
getch();
}
```


Note 1 : When parameterized constructor is defined one must define the default constructor also,otherwise error may occur when a call to default constructor is made.
Note 2: When multiple constructors are defined for a class it is also known as constructor overloading.

Inheritance:-Inheritance is the process of creating a new class from existing class. The existing class isknown as the base/super/parent class and newly created class is known as derived/sub/child class.The derived class will inherit the properties of base class.

Advantages of Inheritance:-

Reusability: It helps the code to be reused in derived class. The base class is defined and once it iscompiled, it needs not to be reworked.

Transitivity: If class B inherits properties of another class A, then all subclasses of class B will automatically inherits the properties of A. It is called transitive property.

Types of Inheritance:

1. Single inheritance:- When a sub class inherits only form one base class, is known as single inheritance.

2. Multiple Inheritance:- When a sub class inherits from multiple base classes, is known as multiple inheritance.

3. Hierarchical Inheritance:- When many sub classes inherit from a single class, it is known as hierarchical inheritance.

4. Multilevel Inheritance:- When a class inherit from a class that itself inherits from another class it is known as a multilevel inheritance.

5. Hybrid Inheritance: It is a combination of two or more of above types of inheritance. There isno pattern of deriving from classes.

Syntax for defining a derived class:

```
class<derived class name>:<visibility mode><base class name>
{
//Data members of derived class
//member functions of derived class
};
```

Visibility modes

The visibility mode in the definition of the derived class specifies whether the features of the base class are privately derived or publicly derived or protected derived.

Constructor and Destructor in Derived classes:

When a base class and a derived class both have constructor and destructor, the constructors are

executed in order of inheritance and destructors are executed in reverse order. That is, the base

constructor is executed before the constructor of the derived class and the destructor of the derivedclass is executed before the base class destructor.

Data File Handling In C++

File: - The information / data stored under a specific name on a storage device, is called a file.

Stream: - It refers to a sequence of bytes.

Text file: - It is a file that stores information in ASCII characters. In text files, each line of text is terminated with a special character known as EOL (End of Line) character or delimiter character.

When this EOL character is read or written, certain internal translations take place.

Binary file:- It is a file that contains information in the same format as it is held in memory. In binary files, no delimiters are used for a line and no translations occur here.

Classes used for different file related operation

ofstream: Object of ofstream class used to write data to the files.
ifstream: Object of ifstream class used to read from files
fstream: Object of fstream class used to both read and write from/to files.
Opening a file
Opening file using constructor

```
ofstreamoutFile("sample.txt"); //output only
ifstreaminFile("sample.txt"); //input only
```

Opening File Using open ()

```
StreamObject.open("filename", [mode]);
ofstreamoutFile;
outFile.open("sample.txt");
ifstreaminFile;
inFile.open("sample.txt");
```

File mode parameter Meaning
ios::app Adds data to the end of file
ios::ate Goes to end of file on opening
ios::binary File opens in binary mode
ios::in Opens file for reading only
ios::out Opens file for writing only
ios::nocreate Open fails if the file does not exist
ios::noreplace Open fails if the file already exist
ios::trunc Deletes the contents of the file if it exist
All these flags can be combined using the bitwise operator OR (|). For example, if we want to openthe file example.dat in binary mode to add data we could do it by the following call to memberfunction open():

```
fstream file;
file.open ("example.dat", ios::out | ios::app | ios::binary);
```

Closing File

```
outFile.close();
inFile.close();
```

Input and output operation
put() and get() function:-
the function put() writes a single character to the associated stream. Similarly, the function get()reads a single character form the associated stream.
example :

```
file.get(ch);
file.put(ch);
```

write() and read() function
write() and read() functions write and read blocks of binary data.
example:

```
file.read((char *)&obj, sizeof(obj));
file.write((char *)&obj, sizeof(obj));
```

Determining End of File.
eof():-returns true (nonzero) if end of file is encountered while reading; otherwise return false(zero).

File Pointers And Their Manipulation

All I/O stream objects have, at least, one internal stream pointer:

ifstream has a pointer known as the get pointer that points to the element to be read in the next inputoperation.

ofstream has a pointer known as the put pointer that points to the location where the next element has to be written. fstream, inherits both, the get and the put pointers.

These internal stream pointers that point to the reading or writing locations within a stream can bemanipulated using the following member functions:

The other prototype for these functions is:

seekg(offset, refposition);

seekp(offset, refposition);

The parameter offset represents the number of bytes(any negative or positive integer value forbackward or forward movement) the file pointer is to be moved from the location specified by theparameter refposition. The refposition takes one of the following three constants defined in the iosclass.

ios::beg start of the file

ios::cur current position of the pointer

ios::end end of the file

Program to count number of words from atext file "input.txt"	Program to count number of vowels in a text file "input.txt"

#include<fstream.h>
void main()
{ ifstream fin;
fin.open("input.txt");
char words[50]; int count=0;
while(!fin.eof())
{ fin>>words;
count++;
}
cout<<"Number of words in file is"<<count;
fin.close();
}
	#include<fstream.h>
	void main()
	{
	ifstream fin;
	fin.open("input.txt");
	char ch; int count=0;
	while(!fin.eof())
	{
	fin.get(ch);
	if(ch=='a'||ch=='e'||ch=='i'||ch=='o'||ch=='u')
	count++;
	}
	cout<<"Number of vowels in file are "<<count;
	fin.close();
	}

seekg() moves get pointer(input) to a specified location

seekp() moves put pointer (output) to a specified location

tellg() gives the current position of the get pointer

tellp() gives the current position of the put pointer

Pointer:- Pointer is a variable that holds a memory address of another variable of same type.

Declaration and Initialization of Pointers :

Syntax :

Datatype *variable_name;

e.g., int *p; float *p1; char *c;

Two special unary operator * and & are used with pointers. The & is a unary operator that returns thememory address of its operand.

e.g., int a = 10; int *p; p = &a;

Pointer arithmetic: Two arithmetic operations, addition and subtraction, may be performed onpointers. When you add 1 to a pointer, you are actually adding the size of whatever the pointer ispointing at. That is, each time a pointer is incremented by 1, it points to the memory location of thenext element of its base type.

e.g. int *p; p++;

If current address of p is 1000, then p++ statement will increase p to 1002, not 1001.

Adding 1 to a pointer actually adds the size of pointer's base type.

Base address : A pointer holds the address of the very first memory location of array where it ispointing to. The address of the first memory location of array is known as BASE ADDRESS.

Dynamic Allocation Operators : C++ dynamic allocation operators allocate memory from the freestore/heap/pool, the pool of unallocated heap memory provided to the program. C++ defines two operatorsnew and delete that perform the task of allocating and freeing memory during runtime.

Pointers and Arrays : C++ treats the name of an array as constant pointer which contains base addressi.e address of first memory location of array.

typedef :- The typedef keyword allows to create alias for data types. the syntax is:

typedefexisting_data_typenew_name ;

e.g. typedefintnum;

Function Overloading: Function overloading is the process of defining and using functions with same name having different argument list and/or different return types. These functions aredifferentiated during the calling process by the number, order and types of arguments passed tothese functions.

Example:

```
int Add (int ,int) ;
double Add (double ,double) ;
float Add (int ,float) ;
```