**What is C++?**  
[C++](http://blog.oureducation.in/c-aptitude-questions-answers/) is a general purpose object oriented programming language invented in the early 1980 by bajarne stroutrup.

**What is class?**  
A class can be declared as a collection of data members along with members function which allows association of data and functions into a single unit called encapsulation.

**What are tehe different features of c++?**  
Following are the different features of the Classes in C++,

* Operators and function overloading
* Free storage management
* Constant types
* References
* Inline function
* Virtual function
* Templates
* Exception handling

**Explain constructor?**  
Constructors is a member function having the same name as that of its class and is executed automatically when the class is instantiated (object is created).

**What is a function?**  
A function is a block of code which executes the statements when we call it.  
It consists of three entities:  
1) the function name.this is simply a unique identifier.  
2) The function parameters.this is a set of zero or more typed identifier.  
3) The function return type this specifies the type of value function returns.

**Explain Inline function?**  
Inline function are those function whose function body is inserted in place of the function call.

**What is function overloading?**  
function polymorphism for function overloading is a concept that allows multiple function to share the same name with different arguments type assigning one or more function function body to the same name is known as function overloading.

**What is implicit and explicit type conversion?**  
In implicit casting in c++ compile automatically handle the type conversion.the final result expressed in the highest precision possible.

Explicit: the conversion of data type of two operands is not automatic but forced.we can force an expression to be a specific type by using a cast.

**What is inheritance in C++ and name the different types of inheritance?**  
it is a technique of organizing information in a hierarchy form. It is like a child inheriting the features of its parent.  
The class which we are inheriting from is called as the base class and the class which inherits called as derived class.

Different types of inheritance are as follows-:  
1) single level  
2) Multi-level  
3) Multiple  
4) Hierarchical(hybrid)  
5) multipath

**What is difference between C and C++ ?**

1. C++ is **Multi-Paradigm** ( not pure OOP, supports both procedural and object oriented) while C follows procedural style programming.
2. In C data security is less, but in C++ you can use modifiers for your class members to make it inaccessible from outside.
3. C follows top-down approach ( solution is created in step by step manner, like each step is processed into details as we proceed ) but C++ follows a bottom-up approach ( where base elements are established first and are linked to make complex solutions ).
4. C++ supports function overloading while C does not support it.
5. C++ allows use of functions in structures, but C does not permit that.
6. **C++ supports reference variables** ( two variables can point to same memory location ). C does not support this.
7. C does not have a built in exception handling framework, though we can emulate it with other mechanism. **C++ directly supports exception handling**, which makes life of developer easy.

**What is a class?**

Class defines a datatype, it's type definition of category of thing(s). But a class actually does not define the data, it just specifies the structure of data. To use them you need to create objects out of the class. Class can be considered as a blueprint of a building, you can not stay inside blueprint of building, you need to construct building(s) out of that plan. You can create any number of buildings from the blueprint, similarly you can create any number of objects from a class.

1. class Vehicle
2. {
3. public:
4. int numberOfTyres;
5. double engineCapacity;
6. void drive(){
7. // code to drive the car
8. }
9. };

**What is an Object/Instance?**

Object is the instance of a class, which is concrete. From the above example, we can create instance of class Vehicle as given below

1. Vehicle vehicleObject;

We can have different objects of the class Vehicle, for example we can have ***Vehicle***objects with 2 tyres, 4tyres etc. Similarly different engine capacities as well.

**What do you mean by C++ access specifiers ?**

Access specifiers are used to define how the members (functions and variables) can be accessed outside the class. There are three access specifiers defined which are***public, private, and protected***

* **private:**  
  Members declared as private are accessible only with in the same class and they cannot be accessed outside the class they are declared.
* **public:**  
  Members declared as public are accessible from any where.
* **protected:**  
  Members declared as protected can not be accessed from outside the class except a child class. This access specifier has significance in the context of inheritance.

**What are the basics concepts of OOP?**

* **Encapsulation**

Encapsulation is the mechanism by which data and associated operations/methods are bound together and thus hide the data from outside world. It's also called data hiding. In c++, encapsulation achieved using the access specifiers (private, public and protected). Data members will be declared as private (thus protecting from direct access from outside) and public methods will be provided to access these data. Consider the below class

1. class Person
2. {
3. private:
4. int age;
5. public:
6. int getAge(){
7. return age;
8. }
9. int setAge(int value){
10. if(value > 0){
11. age = value;
12. }
13. }
14. };

In the class ***Person***, access to the data field age is protected by declaring it as *private* and providing ***public***access methods. What would have happened if there was no access methods and the field age was ***public***? Anybody who has a ***Person***object can set an invalid value (negative or very large value) for the age field. So by encapsulation we can preventing direct access from outside, and thus have complete control, protection and integrity of the data.

* **Data abstraction**

Data abstraction refers to hiding the internal implementations and show only the necessary details to the outside world. In C++ data abstraction is implemented using interfaces and abstract classes.

1. class Stack
2. {
3. public:
4. virtual void push(int)=0;
5. virtual int pop()=0;
6. };
8. class MyStack : public Stack
9. {
10. private:
11. int arrayToHoldData[]; //Holds the data from stack
13. public:
14. void push(int) {
15. // implement push operation using array
16. }
17. int pop(){
18. // implement pop operation using array
19. }
20. };

In the above example, the outside world only need to know about the ***Stack***class and its ***push, pop*** operations. Internally stack can be implemented using arrays or linked lists or queues or anything that you can think of. This means, as long as the push and pop method performs the operations work as expected, you have the freedom to change the internal implementation with out affecting other applications that use your Stack class.

* **Inheritance**

Inheritance allows one class to inherit properties of another class. In other words, inheritance allows one class to be defined in terms of another class.

1. class SymmetricShape
2. {
3. public:
4. int getSize()
5. {
6. return size;
7. }
8. void setSize(int w)
9. {
10. size = w;
11. }
12. protected:
13. int size;
14. };
16. // Derived class
17. class Square: public SymmetricShape
18. {
19. public:
20. int getArea()
21. {
22. return (size \* size);
23. }
24. };

In the above example, class ***Square***inherits the properties and methods of class ***SymmetricShape***. Inheritance is the one of the very important concepts in C++/OOP. It helps to modularise the code, improve reusability and reduces tight coupling between components of the system.

**What are virtual functions and what is its use?** Virtual functions are member functions of class which is declared using keyword 'virtual'. When a base class type reference is initialized using object of sub class type and an overridden method which is declared as virtual is invoked using the base reference, the method in child class object will get invoked.

1. class Base
2. {
3. int a;
4. public:
5. Base()
6. {
7. a = 1;
8. }
9. virtual void method()
10. {
11. cout << a;
12. }
13. };
15. class Child: public Base
16. {
17. int b;
18. public:
19. Child()
20. {
21. b = 2;
22. }
23. virtual void method()
24. {
25. cout << b;
26. }
27. };
29. int main()
30. {
31. Base \*pBase;
32. Child oChild;
33. pBase = &oChild;
34. pBase->method();
35. return 0;
36. }

In the above example even though the method in invoked on Base class reference, method of the child will get invoked since its declared as virtual.

**What do you mean by pure virtual functions in C++? Give an example?**

Pure virtual function is a function which doesn't have an implementation and the same needs to be implemented by the the next immediate non-abstract class. (A class will become an abstract class if there is at-least a single pure virtual function and thus pure virtual functions are used to create interfaces in c++).

**What you mean by early binding and late binding? How it is related to dynamic binding?**

Binding is the process of linking actual address of functions or identifiers to their reference. This happens mainly two times.

* During compilation : This is called early binding: For all the direct function references compiler will replace the reference with actual address of the method.
* At runtime : This is called late binding: In case of virtual function calls using a Base reference, as in shown in the example of question no: 2, compiler does not know which method will get called at run time. In this case compiler will replace the reference with code to get the address of function at runtime. Dynamic binding is another name for late binding.

**What is meant by reference variable in C++?**

In C++, reference variable allows you create an alias (second name) for an already existing variable. A reference variable can be used to access (read/write) the original data. That means, both the variable and reference variable are attached to same memory location. In effect, if you change the value of a variable using reference variable, both will get changed (because both are attached to same memory location).

**What are the difference between reference variables and pointers in C++?**

[This question is usually asked in a twisted way during c++ interviews. Sometimes the interviewer might use examples and ask you to find the error.]

|  |  |
| --- | --- |
| **Pointers** | **Reference Variables** |
| Pointers can be assigned to NULL | References cannot be assigned NULL. It should always be associated with actual memory, not NULL. |
| Pointers can be (re)pointed to any object, at any time, any number of times during the execution. | Reference variables should be initialized with an object when they are created and they cannot be reinitialized to refer to another object |
| Pointer has own memory address and location on stack | Reference variables has location on stack, but shares the same memory location with the object it refer to. |

**Write a program asks the user to enter the length and width of a rectangle. It calculates the rectangle’s area and display the value on the screen**

#include <iostream>

using namespace std;

int main()

{

int length, width, area;

cout << "This progarm calculates the area of a";

cout << " rectangle.\n";

cout << " What is the Length of the regtangle?";

cin >> length;

cout << "What is the width of the rectangle";

cin >> width;

area = length \* width;

cout << "The area of the rectangle is " << area << "\n";

return 0; }

**Write a program to calculate the area of a circle**

#include <iostream>

using namespace std;

int main()

{

const double PI = 3.14159;

double area, radius;

cout << "This program calculates the area of a circle. \n";

cout << " What is the radius of the circle";

cin>>radius;

area = PI \* radius;

cout << "The area is " << area <<endl;

return 0; }

**write different functions of C++ arrays in a program**

1. PRINTING OUT ARRAYS

#include<iostream>

using namespace std;

int main()

{

int a[5]={10,20,30,40,50};

int i;

for(i=0;i<=4;i++)

{

cout<< "value in index i is ="<< a[i]<<endl;

cout

}

}

1. EMPTY ARRAY

#include<iostream>

using namespace std;

int main()

{

int a[5]={};

}

1. INPUTING VALUES IN AN ARRAY

#include<iostream>

using namespace std;

int main()

{

int a[5],i;

cout<< "enter numbers in the array"<<endl;

for(i=1;i<=5;i++)

{

cin>>a[i];

}

}

**write a program showing three variables while checking, miles, and days**

#include <stdio.h>

#include <iostream>

using namespace std;

int get\_sqr(void);

int main()

{

int checking;

unsigned int miles;

long days;

number = 5;

checking = -20;

miles = 4276;

days = 187000;

cout<<"we have made a long journey is" <<checking<<endl;

cout<<"miles\n";

cout<<"our checking account balance is" <<checking;

cout<<"\nAbout"<<days <<"days ago columbus";

cout<<"stood on this spot.\n";

cin.get();

return 0; }

**Write a program to show variable initialization of a leap year**

//#include <stdio.h>

#include <iostream>

//using namespace std;

//int get\_sqr(void);

int main()

{

int month = 2, days = 28;

cout << "Mont" <<month <<"has" <<days <<"days.\n";

cin.get();

return 0; }

**Write a program to calculate hourly wages, including overtimes of an employee**

//#include <stdio.h>

#include <iostream>

//using namespace std;

//int get\_sqr(void);

int main()

{

double regularwages, //to hold regular wages

basePayRate = 18.25 , //base pay rate

regularhours = 40.0, //Hours worked less overtime

overtimewages, //To hold overtime wages

overtimepayrate = 27.78, //Overtime pay rate

overtimehours = 10, //Overtime hours worked

totalwages; //Calculate the regular wages.

regularwages = basePayRate \* regularhours; //Calculate overtime wages

overtimewages = overtimepayrate + overtimehours; //Calculat the total wages.

totalwages = regularwages + overtimewages; //Display the total wages.

cout<<"Wages for this week are $" <<totalwages<<endl;

cin.get();

return 0; }

**When executing a program from the IDE, the console window blinks and then closes immediately. Explain.**

Some compilers (eg. Bloodshed’s Dev C++) don’t automatically pause the console screen after the program has finished executing. If this is the case with your compiler, the following two steps will fix your problem: First, add the following line near the top of your program:

#include <iostream>

Second, add the following code at the end of the main () function (right before the return statement):

std::cin.clear(); // reset any error flags

std::cin.ignore(32767, '\n'); // ignore any characters in the input buffer until we find an enter character

std::cin.get(); // get one more char from the user

This will cause your program to wait for you to press a key before continuing, which will give you time to examine your program’s output before your compiler closes the console window.

such as the commonly suggested system("pause") solution may only work on certain operating systems and should be avoided.Visual Studio will not pause at the end of a console application if it is run with debugging (Debug Menu->Start Debugging). If you want it to pause, you can either use the code solution above, or run your program without debugging (Debug Menu->Start Without Debugging).

**When compiling with Microsoft Visual C++, you get the following error: “c:vcprojectstest.cpp(263) :fatal error C1010: unexpected end of file while looking for precompiled header directive”. Discuss**

This error occurs when the Microsoft Visual C++ compiler is set to use precompiled headers but one (or more) of your C++ code files does not include the stdafx header as the first line. To fix this problem, simply locate the file(s) producing the error (in the above error, test.cpp is the culprit), and add the following line at the very top of the file(s):

#**include "stdafx.h"**

Note that for programs with multiple files, every C++ code file needs to start with this line.

Alternatively, you can turn off precompiled headers.

**When trying to use cin, cout, or endl, the compiler says cin, cout, or endl is an “undeclared identifier”** First, make sure you have included the following line near the top of your file:

#include <iostream>

Second, make sure cin, cout, and endl are prefixed by “std::”. For example:

std::cout << "Hello world!" << std::endl;

**When trying to use endl to end a printed line, the compiler says end1 is an “undeclared identifier”**

Make sure you do not mistake the letter l (lower case L) in endl for the number 1. endl is all letters. I recommend using a font that makes it clear the differences between the letter lower case L, upper case i, and the number 1. Also the letter capital o and the number zero can easily be confused in many non-programming fonts.

**What is the difference between the expression “++a”  and “a++”?**

In the first expression, the increment would happen first on variable a, and the resulting value will be the one to be used. This is also known as a prefix increment. In the second expression, the current value of variable a would the one to be used in an operation, before the value of a itself is incremented. This is also known as postfix increment.

**What would happen to X in this expression: X += 15;  (assuming the value of X is 5)**

X +=15 is a short method of writing X = X + 15, so if the initial value of X is 5, then 5 + 15 = 20.

**In C language, the variables NAME, name, and Name are all the same. TRUE or FALSE?**

FALSE. C language is a case sensitive language. Therefore, NAME, name and Name are three uniquely different variables.

**What is an endless loop?**

An endless loop can mean two things. One is that it was designed to loop continuously until the condition within the loop is met, after which a break function would cause the program to step out of the loop. Another idea of an endless loop is when an incorrect loop condition was written, causing the loop to run erroneously forever. Endless loops are oftentimes referred to as infinite loops.

**What is a program flowchart and how does it help in writing a program?**

A flowchart provides a visual representation of the step by step procedure towards solving a given problem. Flowcharts are made of symbols, with each symbol in the form of different shapes. Each shape may represent a particular entity within the entire program structure, such as a process, a condition, or even an input/output phase.

**What is the difference between functions abs() and fabs()?**

These 2 functions basically perform the same action, which is to get the absolute value of the given value. Abs() is used for integer values, while fabs() is used for floating type numbers. Also, the prototype for abs() is under <stdlib.h>, while fabs() is under <math.h>.

**When is a “switch” statement preferable over an “if” statement?**

The switch statement is best used when dealing with selections based on a single variable or expression. However, switch statements can only evaluate integer and character data types.

**What are global variables and how do you declare them?**

Global variables are variables that can be accessed and manipulated anywhere in the program. To make a variable global, place the variable declaration on the upper portion of the program, just after the preprocessor directives section.

**What are enumerated types?**

Enumerated types allow the programmer to use more meaningful words as values to a variable. Each item in the enumerated type variable is actually associated with a numeric code. For example, one can create an enumerated type variable named DAYS whose values are Monday, Tuesday… Sunday.

**Example 1: Display Multiplication table up to 10**

#include <iostream>

using namespace std;

int main()

{ int n;

cout << "Enter a positive integer: ";

cin >> n;

for (int i = 1; i <= 10; ++i) {

cout << n << " \* " << i << " = " << n \* i << endl;

} return 0;

}

## Example 1: Fibonacci Series up to n number of terms

#include <iostream>

using namespace std;

int main()

{ int n, t1 = 0, t2 = 1, nextTerm = 0;

cout << "Enter the number of terms: ";

cin >> n;

cout << "Fibonacci Series: ";

for (int i = 1; i <= n; ++i)

{ // Prints the first two terms.

if(i == 1)

{ cout << " " << t1;

continue;

} if(i == 2)

{ cout << t2 << " ";

continue;

} nextTerm = t1 + t2;

t1 = t2;

t2 = nextTerm;

cout << nextTerm << " ";

} return 0; }

### Example 1: Compute Power Manually

#include <iostream>

using namespace std;

int main()

{

int exponent;

float base, result = 1;

cout << "Enter base and exponent respectively: ";

cin >> base >> exponent;

cout << base << "^" << exponent << " = ";

while (exponent != 0) {

result \*= base;

--exponent;

}

cout << result;

return 0;

}

**Recursion**

#include<iostream>

using namespace std;

void recursive\_function(int n){

if(n>10) //base case

return;

else{

cout<<"Recursive Function call number "<<n<<endl;

recursive\_function(n=n+1); // here function is calling it self

} } int main(){

int n=1;

recursive\_function(n); // function call

return 0; }

**Right Angle**

#include<iostream>

using namespace std;

int main()

{ for(int i=0;i<=5;i++){

for(int j=0;j<=i;j++)

{ cout<<j;

} cout<<endl;

} return 0;}

**Triangle shapes**

#include<iostream>

using namespace std;

int main()

{ cout<<"\"Hollow Triangle Shape\"\n\n";

int z=1;

for (int i=0; i<7; i++)

{ for (int j=7; j>i; j--)

{ cout<<" "; // displaying space here

} cout<<"\*"; // displaying asterisk here

if (i!=0)

{ for (int k=1; k<=z; k++)

{ cout<<" ";

} cout<<"\*";

z+=2;

} cout<<endl; // endl is for new line

} for (int i=0; i<=z+1; i++)

{ cout<<"\*"; }

return 0; }

**Slanting Rectangle**

#include <iostream>

#include <iomanip>

using namespace std;

void PrintChar(int row = 5, int column = 10, char symbol = '\*');

int main()

{ int rows, columns;

char symbol;

cout << "How many rows and columns do you want, and with what symbol (default is \*) ?" << endl;

cin >> rows >> columns >> symbol;

PrintChar(rows, columns, symbol);

}void PrintChar(int row, int column, char symbol)

{ bbb

for (int y = 1; y <= column; y++)

{ cout << symbol;

return 0; }

**Shapes diamond**

#include<iostream>

#include<conio.h>

using namespace std;

int main()

{ int n, c, k, space = 1;

cout<<"\n\nEnter number of rows: ";

cin>>n; space = n - 1;

for (k = 1; k<=n; k++)

{ for (c = 1; c<=space; c++)

cout<<" ";

space--;

for (c = 1; c<= 2\*k-1; c++)

cout<<"\*";

cout<<"\n";

} space = 1;

for (k = 1; k<= n - 1; k++)

{ for (c = 1; c<= space; c++)

cout<<" ";

space++;

for (c = 1 ; c<= 2\*(n-k)-1; c++)

cout<<"\*";

cout<<"\n"; }

getch(); }

**Loops**

**Case**

#include<iostream>

#include<conio.h>

using namespace std;

int main()

{ int year;

cout<<"enter class year";

cin>>year;

switch (year)

{ case 0: cout<<"not in school"<<endl;

break;

case 1: cout<<"fresh man"<<endl;

break;

case 2: cout<<"sophomore"<<endl;

break;

case 3:cout<<"junior"<<endl;

break;

case 4: cout<<"senior"<<endl;

break;

default: cout<<"career student"<<endl;

} getch();

}

**Right Angle**

#include<iostream>

using namespace std;

int main()

{ for(int i=0;i<=5;i++){

for(int j=0;j<=i;j++)

{ cout<<j;

} cout<<endl;

} return 0;}

**Triangle shapes**

#include<iostream>

using namespace std;

int main()

{ cout<<"\"Hollow Triangle Shape\"\n\n";

int z=1;

for (int i=0; i<7; i++)

{ for (int j=7; j>i; j--)

{ cout<<" "; // displaying space here

} cout<<"\*"; // displaying asterisk here

if (i!=0)

{ for (int k=1; k<=z; k++)

{ cout<<" ";

} cout<<"\*";

z+=2;

} cout<<endl; // endl is for new line

} for (int i=0; i<=z+1; i++)

{ cout<<"\*"; }

return 0; }

**Slanting Rectangle**

#include <iostream>

#include <iomanip>

using namespace std;

void PrintChar(int row = 5, int column = 10, char symbol = '\*');

int main()

{ int rows, columns;

char symbol;

cout << "How many rows and columns do you want, and with what symbol (default is \*) ?" << endl;

cin >> rows >> columns >> symbol;

PrintChar(rows, columns, symbol);

}void PrintChar(int row, int column, char symbol)

{ bbb

for (int y = 1; y <= column; y++)

{ cout << symbol;

return 0; }

**Shapes diamond**

#include<iostream>

#include<conio.h>

using namespace std;

int main()

{ int n, c, k, space = 1;

cout<<"\n\nEnter number of rows: ";

cin>>n; space = n - 1;

for (k = 1; k<=n; k++)

{ for (c = 1; c<=space; c++)

cout<<" ";

space--;

for (c = 1; c<= 2\*k-1; c++)

cout<<"\*";

cout<<"\n";

} space = 1;

for (k = 1; k<= n - 1; k++)

{ for (c = 1; c<= space; c++)

cout<<" ";

space++;

for (c = 1 ; c<= 2\*(n-k)-1; c++)

cout<<"\*";

cout<<"\n"; }

getch(); }

**REVISION QUESTIONS**

1. Discuss the following terms as applied in object oriented programming;
   1. Function ii. Constructor iii. Data members iv. Object
   2. Class vi. Identifier vii. Structure
2. At Jomo Kenyatta University of Agriculture and Technology the admission criteria require that an applicant must got a mean grade equivalent to 60 marks to enroll for Bsc in Computer Science and applicant must have minimum of 60 marks in group I subject; Physics, Mathematics and English. However, a candidate with 65 marks, in groups 2 subject; English, Physics and Chemistry also qualifies for admission.Write an Object Oriented Program that accepts input from an applicant and advise whether one qualifies for admission or not.
3. Write an object oriented program that uses an array to store your names and display the names.
4. If the user enter a negative number, the program should throw an exception, and display a string or message.
5. Discuss function overloading and write aprogram that illustrate the concept of overloading a function
6. Using switch statement write a C++ program to display the day depending ion the choice entered by the user from the key board i.e 1 15 Monday, 2 1 5 Tuesday, 3 Thursday: …….. 7 1 5 Sunday
7. What is an array? With the help of a code show how an array is declared in C++.
8. Explain three (3) reasons as to why object oriented programming is gaining more popularity as compared to procedural programming.
9. Differentiate between local and global variables using a snippet code example.
10. Describe any five data types support by C++ language.
11. Write a C++ program that allows the user to the key in a number of items a customer has bought and the cost of each item the program should display this details should have two function one for input and another display the program should a class called item and two object namely milk and broad.
12. Describe any FOUR types of inheritance used in C++
13. Describe any five characteristics of C++ language as an OOP language.
14. Discuss the syntax of C++ programming language.
15. Briefly explain how to access array in C++ programming.
16. Create a class “my circle” that uses two members function “area” and “circumference” to calculate and print the area and the circumference of a circle plus and additional function “input” to capture the required input from a user. A = , π = 3.142
17. Using appropriate examples discuss types of comment support by C++ language. Using relevant examples write two ways of declaring variable in C++ language.
18. Write a C++ program to calculate the area of a cone.
19. Discuss the syntax of C++ language method.
20. Using appropriate examples, discuss the scope of global and local variable in C++ language.
21. Encapsulation is an important concept in object oriented programming, device a working program that depicts the use of this concept, your program should prompt input from the user.
22. Describe the different ways and how to enforce data abstraction.
23. Discuss the main access specifies in a class declaration.
24. Using a function write a C++ program to multiply two variables and return their sums.
25. Write a C++ program that displays all numbers that are evenly divisible by 7 between 50 and 300 on the standard output console. Use a do...while statement.
26. Highlight any rules and conventions that C++ developers should be adhere to when naming variables, constants and data items.
27. Outline any THREE characteristics of C++
28. Write a program in C++ using for loop to arrange a pattern.
29. Describe the purpose or use of the return statement in a function.
30. Discuss the methods used when declaring constants in C++. Use correct language syntax and sample code snippets in your discussion.
31. Write a program by differentiating between while do and do while loop in C++
32. Distinguish between procedural programming and object oriented programming.
33. Describe a constructor and write an object oriented program that demonstrate the use of constructors.
34. Write a program that requests student’s age from the user. The program should accept positive age only.
35. Write a C++ program that request TWO numbers from the user and calculates the average of the two numbers.
36. Use relevant examples to contrast between the structured and object-oriented programming paradigms. Write a C++ program that implements inheritance.