Welcome to tpspoint.com

Chapter 3

Probable marks : 41

Scope of the Syllabus

- Review of C++
- Arrays, pointers, references, strings
- Principle of object oriented programming
- Classes and objects
- Constructors and destructors
- Operator overloading and type conversions
- Inheritance
- Virtual functions and polymorphism
- Working with files

REVIEW OF C++

Q. 1 What is C++ ? What are the advantages of C++ ? (March 2003, 13

Ans. :

C++ is an object oriented programming language. Initially C++ was named as "C with classes". C++ was developed by Bjarne Stroustrup at AT & T Bell Laboratories, USA, in the early eighties.

The advantages of C++ over C are :

(i) C++ is an incremented version of C. It is a superset of C. Almost all C programs can also run in C++ compiler.

(ii) The important facilities added in C++ are classes, function overloading, operator overloading.

(iii) C++ allow user to create abstract data types, to inherit properties from existing data types.

(iv) C++ supports polymorphism.

(v) Any real life application systems such as editor, compiler, databases, communication systems can be built by C++.

(vi) Object oriented libraries can be built by C++.

(vii) C++ programs can be easily implemented, maintained and expanded.

Q. 2 Differentiate between traditional procedural programming approach and object oriented programming approach. **(Oct. 2002, 2005; March 2011, March 2018)**

Ans. :

The differences between traditional procedural programming approach and object oriented programming approach are as follows :

	Traditional Procedural Programming Approach		Object Oriented Programming Approach
1	In this approach, the problem is viewed as a sequence of things to be done.	1	In this approach, the problem is decomposed into a number of entities called objects and then builds data and function around these entities.
2	Emphasis is on doing things.	2	Emphasis is on the data rather than procedure.
3	Large programs are divided into smaller programs known as functions.	3	Programs are divided into entities known as objects.
4	Data move openly around the system from	4	Data is hidden and cannot be accessed by external functions.
5	Employs top-down approach in program design.	5	Follows bottom-up approach in program design.

Q. 3 What do you mean by Object Based Programming Language and Object Oriented Programming Language ? State the relationship between these languages. **(Oct. 2008, March 2010, 3)**

Ans :

Object Based Programming Language :

1) Language that supports programming with objects are said to be object based programming languages.
2) It is a style of programming that primarily supports encapsulation & object or identity.
3) Major features are :
   a) Data encapsulation
   b) Data hiding & access mechanism
   c) Automatic initalization & clean-up objects
   d) operator overloading.
4) They do not support inheritance .& dynamic binding.
5) For eg-Ada.

Object - Oriented Programming Language :

1) This language incorporates all the object based features along with inheritance and dynamic binding.
2) For eg. — C++, Smalltalk.

The relation between them characterized by following statement :

Object oriented programming =. object—based features + inheritance + dynamic binding

Q. 4 State any six principal advantages of Object Oriented programming. (March 2012, 3)

Ans.:

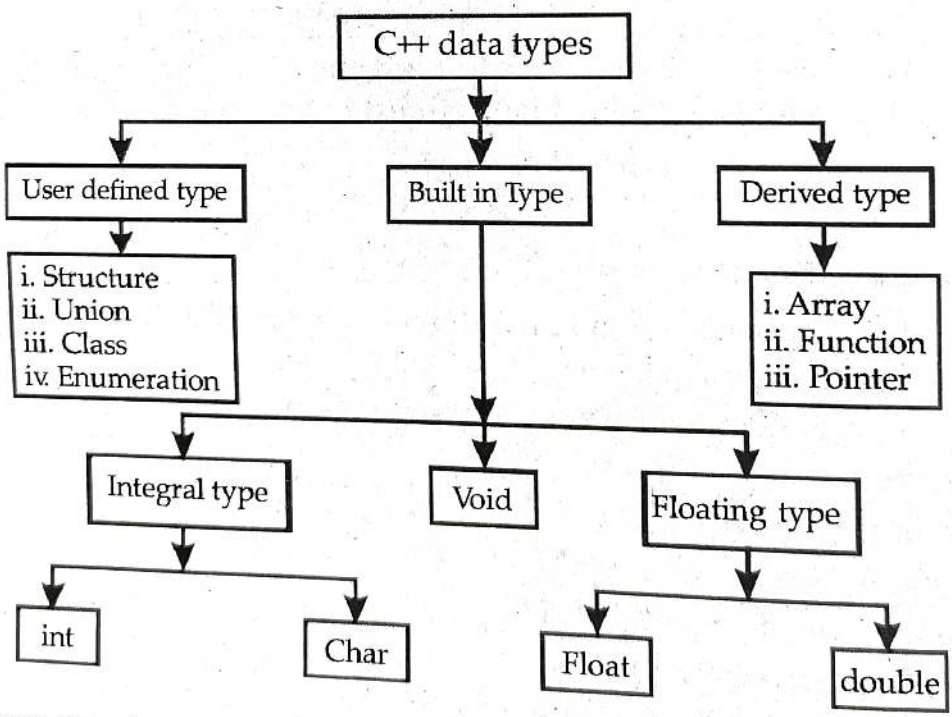
Advantage of Object Oriented Programming:

1. Through, inheritance, eliminate redundant code and extend the use of existing classes.
2. The principle of data hiding helps the programmer to build secure program.
3. It is possible. to have multiple instances of ,an object to co-exist without any interference.
4. It is easy to partition the work in a; project based on object.
5. OOP system can be easily upgraded from small to large system.
6. Software complexity can, be easily managed.
7. It is possible to map objects in the problem domain to objects in the program.
8. Good message passing technique for communication between objects.

Q.5 What are the different data types in C++ ? (Oct.2015)

Ans. :

1) Data types in C++ are shown in figure below :



2) C++ allows user to create new abstract data types, which can behave like any buit-in data type. These are called user-defined data types. These include structure, union, class and enumeration.

3) C++ provides three built-in data types which are integral, void and floating.

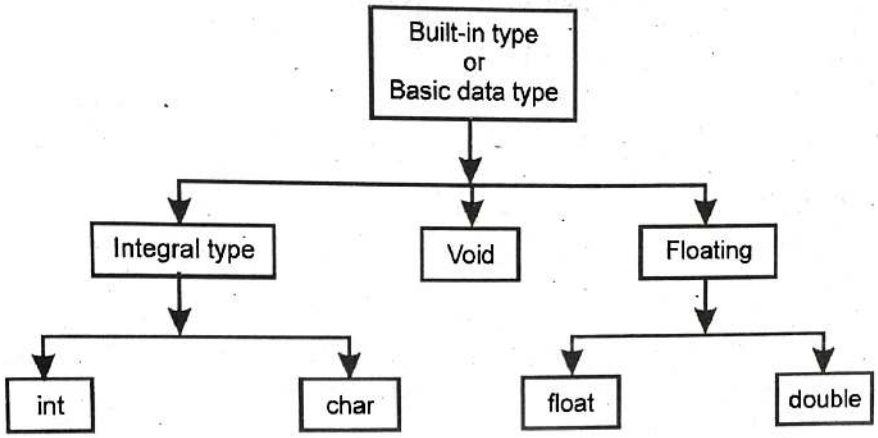
4) Integral includes integer and character (string) while floating type includes float and double.

5) In addition to these data types, C++ provides user with arrays, functions and pointers which are referred as derived data types.

Q. 6 Enlist the basic data types used in C++ with size of data in terms of bytes for each. (March 2002, 2006, October 2006)

OR Enlist different built in data types in C++ with their sizes. (Oct. 2009)

Ans. : There are three main basic built-in data types used in C++ viz. integral type, void and floating type.



i) Integral data type :

It includes integer (int) and character (char).

An int variable requires 2 bytes to store, while a character variable requires 1 byte.

Integer variables are also of two types : (a) short int and (b) long int. Long integer requires 4 bytes, while short integer requires 2 bytes.

ii) Void data type :

Void data type is used :

(a) to specify the return type of a function when it is not returning any value.

(b) to indicate an empty argument list to a function.

(c) to declare generic pointers.

(July 2019)

iii) Floating type :

Floating type variables are of two types; float and double. A float variable requires 4 bytes, while double requires 8 bytes to store in memory.

There is another kind of double namely long double, which requires 10 bytes to store in memory. The following table shows all basic data types, size and range :

Sr. No.	Type	Bytes	Range
1	char (Signed char)	1	– 128 to 127
2	unsigned char	1	0 to 255
3	int (short int or signed int)	2	– 32768 to 32767
4	unsigned int	2	0 to 65535
5	float	4	3.4×10^{-38} to 3.4×10^{-38}
6	double	8	1.7×10^{-308} to 1.7×10^{308}
7	long double	10	3.4×10^{-4932} to 3.4×10^{-4932}

Q. 7 Explain insertion and extraction operators in C++. (Mar. 2012)

Ans. :

(i) Insertion operator :

The operator "<<" is called as insertion operator. It is also called as "put to" operator. It inserts the contents of the variables on its right to the object on its left.

It is generally used in output statement in C++.

e.g. (i) Cout << a ;

(ii) Cout << "program";

In first example, the value of variable 'a' is printed on screen, while in second example the word "program" is printed on screen.

(ii) Extraction operator :

The operator ">>" is called as extraction operator. It is also called as 'get from' operator. It extracts or takes the value from keyboard and assigns it to a variable on its right. It is used in input statement in C++.

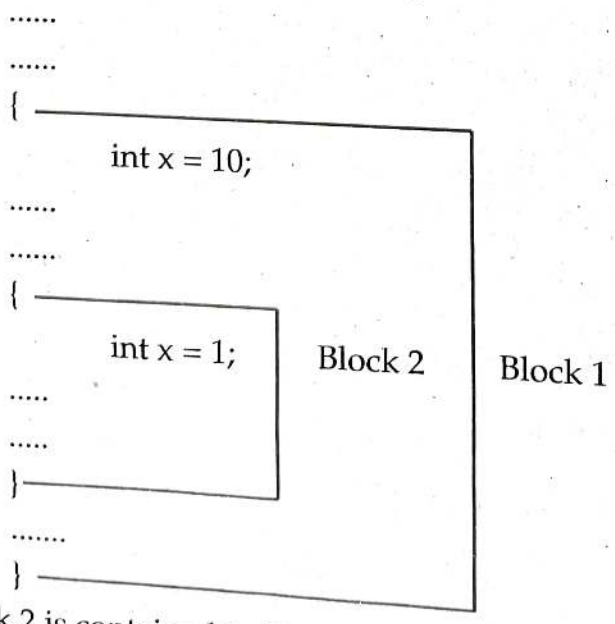
e.g. cin >> a;

This instruction will extract a value from keyboard and assign it to the variable a. C++ allows us to redefine insertion and extraction operators by overloading them.

Q. 8 Write a short note on scope resolution operator. (Oct. 2014; Mar.16)

Ans. :

1) The operator :: is called as scope resolution operator.
2) C++ is a block structured language i.e. a C++ program may contain one block within another block.
3) When a variable is declared in program, scope extends from the point of declaration till the end of the block in which it is defined.
4) The same variable name can be used to have different meaning in different blocks.
5) Consider the following segment of program.



Here Block 2 is contained in Block 1. Note that declaration of a variable in an inner block hides the declaration of the same variable in an outer block.

6) Scope resolution operator is used to uncover a hidden variable.
It takes the form

:: variable name

e.g.

```
.....
.....
{
    int x = 10;
    .....
    {
    int x = 1;
    cout << "Local x is" << x;
    cout << "\n Global x is" << :: x;
.....
}
.....
}
```

The output will be as follows :
Local x is 1
Global x is 10

Q. 9 Explain the use of scope resolution operator and memory management operators in C++ with examples. (March 2004,16, 17)

Ans. : Scope resolution operator :

1) In C++, scope resolution operator (: :) is used to access a global variable from a function in which a local variable is defined with the same name as a global variable.

2) For example :
In following program, the function main ( ) access the global variable num and also the local variable with the same name.

```
int num = 20
void main ( )
{
    int num = 10; // local variable
    cout <<"Local =" << num;
    cout <<"Global =" << : : num;
}
```

The output is as :
Local = 10
Global = 20

Memory management operator :

(1) C++ provides following two memory management operator :
(i) new (ii) delete

(2) The new operator obtains memory block from operating system and returns a pointer to its starting point. The new operator returns NULL, if memory allocation is unsuccessful.
The general format of new operator is :

DataType * new DataType [size in integer];

(3) The delete operator is used to return the memory allocated by the new operator back to the memory pool. Thus released memory will be reused by other parts of the program.
The general format of delete operator is :
delete pointervariable;

(4) For example :

```
void main ( )
{   char * str = "COMPUTER";
    int len = strlen (str);
    char * ptr;
    ptr = new char [len + 1];
    strcpy (ptr, str);
    cout << "ptr =" << ptr;
    delete ptr;
}
```

In above example, the new operator returns a pointer that point to a memory section large enough to hold the string str plus an extra byte for null character. Then after use of memory delete operator released memory.

Q. 10 What are the different selection (conditional) statements in C++? Give syntax for each.

Ans.

The program has to be able to evaluate conditions and select alternative path in program.

In C++, there are two ways in which selection may be made :

1) The if statement 2) The switch statement

1) The if statement :

The if statement has two forms :

i) Simple if statement ii) if _____ else statement

i) Simple if statement :

Syntax :

```
if (condition)
{
    action 1;
}
action 2;
```

1) Depending on the condition value, program execution proceeds in one direction or another.
2) If the condition is true, then action 1 will be done.

ii) if .... else statement :

Syntax :

```
if (condition)
{
action 1;
}
else
{
action 2;
}
action 3;
```

If the condition is true, then and then only action 1 will be done, otherwise action 2 will be done.

2) The switch statement :

1) This is a multiple branching statement.
2) Depending on certain condition, it executes only one module out of several. If no condition is satisfied, then default module will be executed.
3) The break statement is used to terminate switch statement.
4) Expression must have int or char value.

Syntax :

```
switch (expression)
{
case 1 :
    {
        action 1;
        break;
    }
case 2 :
    {
        action 2;
        break;
    }
:
:
default :
    {
    action x;
    }
}
```

Q. 11 What are the different looping structures in C++ ? Give syntax for each.

Ans. :

Following are the different looping structures in C++ :

1) For loop 2) While loop 3) Do-while loop

1) The for loop :

The for loop is an entry-controlled loop. It is used when action is to be repeated predetermined number of times.

Syntax :

```
for (initial expression; test expression; increment / decrement expression)
{ .....
action;
} .....
```

where :

(a) Initial expression is executed only once, when the loop starts.

(b) Test-expression evaluated each time through the loop, before the body of the loop is executed.

(c) Increment / Decrement expression changes the value of the loop variable at the end of the loop.

2) The while loop :

The while loop is an entry-controlled loop and it repeats the action until the condition becomes false. When condition is false, that time loop is terminated.

Syntax :

```
while (condition)
{
        action 1;
}
        action 2;
```

3) The do-while loop :

The do-while loop is an exit-control loop used to carry out conditional looping.

Syntax :

```
do
{
        action 1;
}
        while (condition);
        action 2;
```

In do-while, condition is not tested until the body of the loop has been executed once. So even if the condition is false the loop is executed at least once. If the condition is false after the first iteration, the loop is terminated.

Q. 12 What is function prototyping ?

Ans. :

1) Function prototyping is one of the major improvements added to C++ functions.
2) The prototype describes the function interface to the compiler by giving details such as the number and the type of arguments and the type of return values.
3) With function prototyping, a template is always used when declaring and defining a function.
4) When a function is called, the compiler uses the template to ensure that proper arguments are passed, and the return value is treated correctly.
5) Any violation in matching the arguments and the return type will be caught by the compiler at the time of compilation itself.
6) Function prototype is a declaration statement in the calling program and is of the following form

```
return-type function-name (argument-list);
```

The argument list contains the types and names of arguments that must be passed to the function.

e.g.

float volume (int x, float y, float z);

Note that each argument variable should be declared independently. The combined declaration like :

float volume (int x, float y, z); is invalid.

7) In function declaration, the names of arguments are the dummy variables and therefore, they are optional i.e. the declaration :-

float volume (int, float, float); is valid.

Q. 13 Write a short note on inline functions.

Ans. :

1) When a function is called, a lot of time is spent in executing a series of instructions, for tasks such as jumping to the function, saving registers, pushing arguments into stack and returning to the calling function.
2) C++ proposes a solution of inline functions to this problem. Inline function makes a program run faster because the overhead of a function call and return is eliminated.
3) However, it makes program to take up more memory, because the statements that define inline function are reproduced at each point where the function is called.
4) "An inline function is a function that is expanded inline when it is invoked". i.e. the compiler replaces function call with the corresponding function code.
5) The inline functions are defined as follows :

```
inline function header
{
function body
}
```

```
e.g.  inline int area (int a, int b)
      {
      return (a*b);
      }
```

6) The functions are generally made inline, when they are small enough to be defined in one or two lines.
7) The keyword inline is not a command, but it is a request to the compiler.
8) Following are some situations in which compiler may ignore inline request :
   i) For functions returning value, if loop, switch or goto statement exists.
   ii) For functions not returning value, if a return statement exists.
   iii) If functions contain static variables.
   iv) If inline functions are recursive.

Q. 14 What are default arguments? Give the advantages of using default arguments.

Ans. :

1) C++ allows to call a function without specifying all its arguments. In such cases, the function assigns a default value to the parameter, which does not have a matching argument in the function call.
2) Default values are specified when the function is declared.
3) Consider a function area declared as follows,

   float area (int r, float Pi = 3.14);

   The above prototype declares default value 3.14 to the argument Pi. A subsequent function call like -

   A = area(7); //one argument missing

   passes the value 7 to r and lets the function use default value 3.14 for Pi.
   The call A = area (7, 2.5) passes an explicit value 2.5 to Pi.
4) Only trailing arguments can have default values. i.e. add defaults from right to left. A default value cannot provide to an argument in the middle of list.
5) Advantages of using default arguments :
   i) These are useful in situations, where some arguments have same values.
   ii) It provides better flexibility to programmers by allowing to use particular arguments that are meaningful to particular solution.
   iii) Use default arguments to add new parameters to the existing functions.
   iv) Default arguments can be used to combine similar functions into a single function.

Q. 15 Explain the concept of function overloading with example.

(March 2008, 15, 17; Oct. 2008)

Ans. :

1) The use of same function name to create functions that perform a variety of different tasks is called as function overloading.
2) Overloading refers to the use of same thing for different purposes. Function overloading or function polymorphism, is an example of compile time polymorphism.

3) Using the concept of function overloading, create a family of functions with one function name but with different argument lists.

4) The function would perform different operation, depending on argument list in function call.

5) The correct function to be invoked is determined by checking the number and the type of the arguments and not on the function type.

6) e.g.

```
# include <iostream.h>
int area (int s);                    //prototype declaration
int area (int l, int b);             //for overloading area( )
main ( )
{
cout <<area (10);                    //function calls
cout <<area (5, 10);
}
int area (int s)                     //function definition
{
return (s*s);
}
int area (int l, int b)
{
return (l*b);
}
```

In above example the function area( ) is overloaded. The first function is used to calculate area of square. It has one integer parameter.

The second function is used to calculate area of rectangle. It has two integer parameters.

7) When a function is called, the compiler first matches the prototype having same number and types of arguments and then calls appropriate function for execution. A best match must be unique.

Q. 16 Explain the structure of a general C++ program. **(March 2019)**

Ans. :

1) A typical C++ program contains 4 sections as shown in following figure These sections may be placed in different code files and then compiled independently or jointly.

Include files
Class declaration
Class functions definitions
Main function program

Structure of C++ program

2) It is a common practice to organize a program into three separate files.

3) The class declarations are placed in a header file and the definitions of the member go in other file.

4) This approach enables the programmer to separate the abstract of the interface from the implementation details.

5) Finally the main program that uses the class is placed in third file, which includes the previous two files as well as any other files required.

Q. 17 Write a program in C++ that finds larger number among three numbers.

Ans. :

```
//Program to find largest number
#include <iostream.h>
void main( )
{
int a, b, c, max;
cout<<"Enter three numbers" <<endl;
cin>>a>>b>>c;
if (b>c)
        {max = b;}
else
        {max = c;}
if (a>max)
        {max = a;}
cout<<"The larger number is:-";
cout<<max;
}
```

Q. 18 Write a program in C++ to display a fibonacci series of 15 terms.

(March 2004, 2007, 2009, 2017; Oct. 2002)

OR Write a program in C++ to display a Fibonacci series of 20 terms (use n < = 18 in this case)

Ans. :

```
#include<iostream.h>
void main( )
{
int f0, f1, f, n;
f0 = 0;
f1 = 1;
clrscr( );
cout<<"Fibbonacci series\n";
cout<<"\n" <<f0<<"\n"<<f1;
for (n=1 ; n<=13; n++)
{
        f = f0 + f1;
        cout<<"\n"<<f;
        f0 = f1;
        f1 = f;
}
}
```

Q. 19 Write a program in C++ to calculate and print factorial of first 10 numbers.

Ans. :

```
//C++ program to calculate and print factorial of first 10 numbers
#include<iostream.h>
#include<conio.h>
void main( )
{
        int fact, n, i;
        clrscr( );
        cout<<"Number"<<"\t"<<"Factorial";
        for (n=1; n<=10; n++)
        {
                fact = 1;
                for (i = 1; i<=n; i++)
                {
                fact = fact*i;
                }
        cout<<endl<<n<<"\t"<<fact;
        }
}
```

Q. 20 Write a C++ program to find factorial of a natural number input during program execution. (March 2004, 08, 17, Oct. 2002,04,12)

Ans. :

```
//Program to find factorial of a number
#include<iostream.h>
#include<conio.h>
void main ( )
{
        int fact, numberi;
        clrscr ( );
        fact = 1;
        cout << "Enter the number" <<endl;
        cin >> number;
        for (i = 1; i < = number; i++)
        {
        fact = fact * i;
        }
        cout << "The factorial of a inputted number is :" << fact;
}
```

Q. 21 Write a program in C++ to check whether the given integer is palindrome or not. (Oct.2012)

Ans. :

```
//C++ program to find whether the given integer is palindrome or not
#include<iostream.h>
#include<conio.h>
```

```
void main( )
{
int n, dn, temp=0, d;
clrscr( );
cout<<"Enter a number";
cin>>n;
dn=n;
while (dn!=0)
{
        d=dn%10;
        temp=(temp*10)+d;
        dn=dn/10;
}
if (n==temp)
{
cout<<"The number"<<n<<"is palindrome";
}
else
{
cout<<"The number"<<n<<"is not palindrome";
}
}
```

Q. 22 What is an armstrong number ? Write a program in C++ to check whether the give number is armstrong or not.

Ans. : Armstrong number :

"If sum of the cubes of digits of a number is equal to the original number, then th number is said to be an armstrong number".

e.g. 153 is an armstrong number.

```
//C++ Program to find whether the number is armstrong or not.
#include<iostream.h>
#include<conio.h>
void main( )
{
int n, dn, temp, d;
cout<<"Enter a number";
cin>>n;
dn = n;
temp=0;
while (dn!=0)
{
        d=dn%10;
```

```
        temp=temp+(d*d*d);
        dn=dn/10;
    }
    if(n==temp)
    {
    cout<<n<<"is armstrong no.";
    }
    else
        {
        cout<<n<<"is not an armstrong number";
        }
    }
```

Q. 23 Write a program in C++ to print the numbers in following manner.

1
2 2
3 3 3
4 4 4 4
:
:
n terms

Ans. :

```
//C++ program to print given pattern
#include<iostream.h>
#include<conio.h>
void main( )
{
int i, j, n;
clrscr( );
cout<<"Enter a number";
cin>>n;
cout<<endl;
for (i=1; i<=n; i++)
{
        for (j=1; j<=i; j++)
        {
                cout<<i<<"\t";
        }
        cout<<endl;
}
}
```

Q. 24 Write a program in C++ to print the numbers in following manner.

```
1  0  1  0  1
1  0  1  0
1  0  1
1  0
1
```

Ans. :

```
//C++ program to print given pattern
#include<iostream.h>
#include<conio.h>
void main( )
{
clrscr( );
int i, j;
        for (i=5; i>=1; i- -)
        {
                for (j=1; j<=i; j++)
                {
                        cout<<j%2<<"\t";
                }
                cout<<endl;
                }
        }
```

Q. 25 Write a program to perform arithmetic calculations such as addition, subtraction multiplication or division, depending on choice using switch statement.

Ans. :

```
//C++ program to generate simple calculator
#include<iostream.h>
#include<conio.h>
void main( )
{
float a, b, result;
int ch;
clrscr();
cout<<"Enter two numbers";
cin>>a>>b;
cout<<"\n1-addition\n 2-subtraction\n 3-multiplication \n 4-division";
cout<<"Enter Your Choice :";
cin>>ch;
switch (ch)
{
```

```
        case 1:
                result=a+b;
                cout<<"Sum is"<<result;
                break;
        case 2:
                result=a-b;
                cout<<"Difference is"<<result;
                break;
        case 3:
                result=a*b;
                cout<<"Product is"<<result;
                break;
        case 4:
                result=a/b;
                cout<<"Division is"<<result;
                break;
        default:
                cout<<"invalid choice";
                break;
        }
}
```

Q. 26 **What is a recursive function ? Write a program in C++ to calculate addition of first n numbers using recursive function.**

Ans. : **Recursive function :**

"A function which is called within the body of the same function itself is called as recursive function."

```
//Program to calculate addition of first n numbers
        #include<iostream.h>
        #include<conio.h>
        int add (int);
        void main( )
        {
        int n, sum;
        cout<<"Enter a number \n";
        cin>>n;
        sum=add(n);
        cout<<"Addition of first"<<n<<"numbers is"<<sum;
        }
        //function to calculate addition
        int add (int x)
        {
        int S = 0;
        if (x!=0)
```

```
{
S=x+add(x-1);
}
return(S);
}
```

Q. 27 Write a program in C++ to calculate volume of cube, cylinder and rectangular box depending on choice by using function overloading.

Ans. :

```
//Program using function overloading
#include<conio.h>
#include<iostream.h>
int volume (int s);
float volume (float r, float h);
int volume (int l, int b, int h);
void main( )
{
int ch;
do
{
        clrscr( );
        cout<<"\n1:Volume of cube";
        cout<<"n2:Volume of cylinder";
        cout<<"\n3:Volume of rectangular box";
        cout<<"\n4:Quit";
        cout<<"\n\n Enter Your Choice";
        cin>>ch;
switch (ch)
{
case 1:
        cout<<"Volume of cube is";
        cout<<volume (5);
        break;
case 2:
        cout<<"Volume of cylinder is";
        cout<<volume (7.0, 2.0);
        break;
case 3:
        cout<<"Volume of rectangular box is";
        cout<<volume (3, 5, 7);
        break;
```

```
case 4: break;
default :
        cout<<"Invalid choice";
        cout<<"Reenter your choice";
}
}
        while (ch!=4);
}
        //function to calculate volume of cube
        int volume (int s)
{
        return (s*s*s);
}
        //function to calculate volume of cylinder
        float volume (float r, float h)
{
        return (3.14*r*r*h);
}
        //function to calculate volume of rectangular
        int volume (int l, int b, int h)
{
        return (l*b*h);
}
```

Arrays, Pointers, References and Strings

Q. 28 What is an array ? Explain how array can be passed onto a function.

Ans. :

1) "An array is a collection of identical data objects, which are stored in consecutive memory locations under common variable name."
2) Arrays may be one dimensional or multidimensional.
3) The general form for declaration of one-dimensional array is

```
data-type array-name [expression];
```

e.g. int a[10];

This declaration creates an array of 10 integers.

4) In general, C++ arrays are zero based. i.e. in above examples, the first array element has index 0 and it is referred as a[0]. Similarly, second array element is a[1] and the last i.e. 10th element is a[9].

5) C++ allows to pass the entire array onto a function.
An array name can be used as an argument for the function declaration. No subscript brackets are required to invoke a function using arrays.
e.g. float rev (float b[], int c);
This declares a function rev, with two parameters, out of which one is an array.

Q. 29 What are pointers ? Give the advantages of using pointers. (March 11, 19, July 16, 17)

Ans. :

1) "A pointer is a variable, which holds the memory address of other variable."
2) * operator is used to declare pointer in C++. It takes the form as :

datatype * variable name;

e.g. int *ptr;
The above declaration will create a variable ptr, which is a pointer variable and which will point to a variable, whose data type is integer.
3) The data type of ptr is not integer, but data type of variable which ptr will point is integer.
4) Advantages of using pointers are as :
   i) It allows to pass variables, arrays, functions, strings, structures, objects as function arguments.
   ii) It allows to return structured variables from functions.
   iii) It supports dynamic allocation and deallocation of memory segments.
   iv) By using pointers, variables can be swapped, without physically moving them.
   v) It allows to establish link between data elements or objects.

Q. 30 What are pointers in C++ ? Explain the use of pointer variables for function definitions using call by value and call by reference OR

(March 2004,07,08,09, Oct. 2006,11)

Explain 'Call by value' and 'Call by reference' with one example of each.

Ans. :

1) Pointers in C++ :
A pointer is a variable which holds the memory address of another variable.
* operator is used to declare pointer in C++.
For example : int *ptr;
where ptr is a pointer variable and which will point to a variable whose data type is integer.

2) The use of pointers in a function definition may be classified into two groups :
(1) Call by value (2) Call by reference.

3) Call by value :
   (a) When a portion of the program invokes a function, control will be transferred from the main function to the calling function and the value of actual arguments is copied to the function.
   (b) Within function the actual value may be altered or changed.

(c) When the control is transferred back from function to the program, altered values are not transferred back. This type of passing formal argument to a function is called as call by value

(d) For example :

```
main ( )
{ void funct (int X, int Y);
        .....
funct (X, Y); // Call by value
.....
}
void funct (int a, int b)
{ .....
}
```

4) Call by reference : (Oct.2014)

(a) In call by reference, when a function is called by a program the address of the actual arguments are copied on to the formal arguments. i.e. the formal and actual arguments are referring to same memory location.

(b) Therefore change in value of formal argument affects the value of actual arguments.

(c) The content of a variable that are altered within the function are return to calling portion of a program in the altered form.

(d) For example :

```
main ( )
{
void funct (int * X, int * Y);
.....
.....
funct (&X, &Y); // Call by reference.
.....
}
void funct (int * a, int * b)
{
.....
}
```

Q. 31 Explain how the memory address of a variable can be accessed in C++. (March 2004, 07,14 ; Oct. 2004,12)

Ans. :

1) Computer uses memory for storing the values of variables and the memory is a sequential collection of storage cell. Each cell has a number called address of the cell.

2) In C++, if declare a variable, then it gets associated with certain location where the value of the variable is stored.

3) Consider the declaration

int p = 30;

then p → Location name (variable)

30 → Value at location

7940 → Location number (address)

Computer has selected 7940 memory location to store the value 30.

4) To access the memory address of a particular variable '&' operator is used. The '&' operator returns the memory address of its operand.

For example : a = &p;

assigns the memory address of variable p to the a. This address is the location address of variable. The operator '&' is "the address of" operator.

The variable 'a' is declared as pointer variable as that contains the address. The pointer variable declared in C++ as,

int * a;

where * indicates that a is a pointer variable.

Q. 32 What is call by reference ? What is the advantage of call by reference over call by value ? (Mar. 2014)

Ans. :

A function can be called by two methods :

(i) Call by value

(ii) Call by reference

1) When a function call passes arguments by value (call by value) the called function creates a new set of variables and copies the values of arguments into them.

2) The function does not have access to the actual variables in the calling program and can only work on the copies of values.

3) Provision of reference variables in C++ permits to pass parameters to the function by reference.

4) When pass arguments by reference (call by reference) the formal arguments in the called function become alises to the actual arguments in the calling function. This means that when the function is working with its own arguments, it is actually working on the original data.

5) The mechanism of call by value is good, if the function does not need to alter the values of the original variables in the calling program.

6) But, if a situation to change the values of variables in the calling program. e.g. in bubble sort compare two adjacent elements in the list and interchange them if first is greater than second. In such situation, the function should be able to interchange the values of variables of calling program, which is not possible by call by value. But it can be done if the call by reference method is used.

7) e.g. //Program to interchange the values of variable

```
#include <iostream.h>
void swap (int*, int*); //function declaration
```

```
void main( )
{
        int a, b;
        cin>>a>>b;
        swap (&a,&b); //call by reference
        cout<<"a="<<a;
        cout<<"b="<<b;
}
void swap (int*a, int*b) //function definition
{
        int t;
        t=*a; //assign the value at address a to t.
        *a=*b; //put the value at b into a.
        *b=t; //put the value at t into b.
}
```

Q. 33 Explain Library Functions : (Oct. 2015, 4)
(i) Strcpy ( ) (ii) Strcmp ( )

Ans. :

(i) Strcpy ( )

If S_1 and S_2 are string then strepy (S_1, S_2) copies character string S_2 into character string S_1. It means it creates a duplicate of string S_2.

Char * strcpy (Char * S_1, const char * S_2)

For example

```
int main ( )
{
char S1 [] = "ABCD"
char S2 [] = "XYZ"
cout << "Before strcpy (S1, S2) \ n";
cout << "\t S1 = [" << S1 <<"], length = " << streln (S1) << endl;
cout << "\t S2 = [" << S2 <<"], length = " << strln (S2) << endl;
strcpy (S1, S2);
cout <<"After strcpy (S1, S2) \n";
cout << "\t S2 = [" << S2 <<"], length = " << strlen (S2) << endl;
}
```

O/P-Before strcpy (S_1, S_2)

S_1 = [ABCDE], length = 5

S_2 = [XYZ]; length = 3

After strcpy (S_1, S_2)

S_1 = [XYZ], length = 3

S_2 = [XYZ], length = 3

(ii) Strcmp ( ) :

Int strcmp (char * S_1, char * S_2);

It compares S_1 with S_2 Returns a negative integer, zero or positive integer according to S_1 is less than equal to or grater than S_2.

Example :

Char *S_1 = "ABCDE"

Char * S_2 = " "

If (strcmp (S_1, S_2) <0)

Cout << S_1 << "<" <<S_2 << endl;

else

cout <S_1<< "> = " << S_2 << endl;

O/P is → ABCDE > =

Q. 34 Write a C++ program to read 'n' numbers input from keyboard and sort them in ascending order. (Oct. 2002)

Ans. :

```
//Program to sort numbers in ascending order
#include <iostream.h>
voide main ( )
{
    int a [100];
    int n, i, j, temp;
    cout << "How many numbers ?" <<endl;
    cin >> n;
    cout << "Enter the elements :";
    for (i=1; i<=n; i++)
    {
        cin >> a[i];
    }
    for (i=1; i<=(n–1); i++)
    {
        for (j=1; j<=n–i; j++)
        {
            if (a[j] > a[j+1])
            {
                temp = a[j];
                a[j] = a[j+1];
                a[j+1] = temp;
            }
        }
    }
    cout<<"Ascending order is :\n";
```

Thank you for visiting
tpspoint.com