FUNCTIONS IN C++

FUNCTIONS

```
void show(); /* Function declaration */

void show() /*Function definition */
{
    /* Function body */
}

int main()
{
    show(); /* Function call */
    return 0;
}
```

MAIN FUNCTION

```
int main(int argc, char * argv[])
{
    return 0;
}

int main()
{
    return 0;
}
```

FUNCTION PROTOTYPING

```
Type functionName(list of arguments)
          float volume(int x, float y , float z);

Without the dummy argument names
          float volume(int , float , float );

Empty parenthesis
          void show(void);
Or
          void show();
Open parameter list
          #include <cstdarg>
          void show(int count, ...)
```

CALL BY REFERENCE

Formal arguments of the function are the aliases of the actual arguments in the calling function

```
void swap(int &a, int &b)
{
    int t = a;
    a = b;
    b = t;
}
```

INLINE FUNCTIONS

- To eliminate the cost of calls to small functions C++ proposes a new feature called inline function.
- An inline function is a function that is expanded inline when it is invoked .That is the compiler replaces the function call with the corresponding function code.
- Syntax

```
    inline function-header
    {
            function body;
    }
```

- Example

```
inline double cube (double a)
{
    return(a*a*a);
}
```

- Some of the situations where inline expansion may not work are:
- 1. For functions returning values if a loop, a switch or a go to exists.
- 2. for function s not returning values, if a return statement exists.
- 3. if functions contain static variables.
- 4. if inline functions are recursive,.

DEFAULT ARGUMENTS

- C++ allows us to call a function with out specifying all its arguments.
- In such cases, the function assigns a default value to the parameter which does not have a matching arguments in the function call.
- Default values are specified when the function is declared .
- The compiler looks at the prototype to see how many arguments a function uses and alerts the program for possible default values.
- Example:
  - `float amount (float principle, int period ,float rate=0.15);`

CONST ARGUMENTS

- In C++, an argument to a function can be declared as unit as const as shown below.

```
int strlen(const char *p);
int length(const string &s);
```

- The qualifier const tells the compiler that the function should not modify the argument
- The compiler will generate an error when this condition is violated .This type of declaration is significant only when we pass arguments by reference or pointers.

RECURSION

- A situation when a function calls itself
- One statement in the function definition makes a call to the same function in which it is present.
- Just as a loop has a conditional check to take the program control out of the loop, a recursive function also posses a base case which returns program control from the current instance of the function to the calling function

FUNCTION OVERLOADING

- Overloading refers to the use of the same thing for different purposes .
- C++ also permits overloading functions .This means that we can use the same function name to creates functions that perform a variety of different tasks.
- This is known as function polymorphism in oops.
- Using the concepts of function overloading , a family of functions with one function name but with different argument lists in the functions call .
- The correct function to be invoked is determined by checking the number and type of the arguments but not on the function return type.

FUNCTION OVERLOADING

- The compiler first tries to find an exact match in which the types of actual arguments are the same and use that function .
- If an exact match is not found the compiler uses the integral promotions to the actual arguments such as :
  - char to int
  - float to double
- to find a match
- When either of them tails ,the compiler tries to use the built in conversions to the actual arguments and them uses the function whose match is unique .
- If the conversion is possible to have multiple matches, then the compiler will give error message.
- Example:
  - `long square (long n);`
  - `double square(double x);`
- A function call such as :- `square(10);`
- Will cause an error because int argument can be converted to either long or double .There by creating an ambiguous situation as to which version of square( )should be used.

MATH LIBRARY FUNCTION

ceil(x)	Returns the value of x rounded up to its nearest integer
cos(x)	Returns the cosine of x
cosh(x)	Returns the hyperbolic cosine of x
exp(x)	Returns the value of E^x
expm1(x)	Returns e^x -1
fabs(x)	Returns the absolute value of a floating x
fdim(x, y)	Returns the positive difference between x and y
floor(x)	Returns the value of x rounded down to its nearest integer
hypot(x, y)	Returns sqrt(x^2 +y^2) without intermediate overflow or underflow
fma(x, y, z)	Returns x*y+z without losing precision
fmax(x, y)	Returns the highest value of a floating x and y
fmin(x, y)	Returns the lowest value of a floating x and y
fmod(x, y)	Returns the floating point remainder of x/y
pow(x, y)	Returns the value of x to the power of y
sin(x)	Returns the sine of x (x is in radians)
sinh(x)	Returns the hyperbolic sine of a double value
tan(x)	Returns the tangent of an angle
tanh(x)	Returns the hyperbolic tangent of a double value