15

FUNCTIONS

- **FUNCTIONS:** Functions are blocks of code that perform a task to accomplish something productive. When you want to group same statements depending upon their actions like finding total, average, etc., you can put them in a function.
- A function may be user defined function or library function.
- To use library functions in a C++ program you have to include header files. A header file may be included in one of the two ways. # include or # include "iostream.h". The header file in angle brackets means that file reside in standard include directory. The header file in double quotes means that file resides in current directory.
- **LIBRARY FUNCTIONS:**

1. MATHEMATICAL FUNCTIONS:

Function	Header File	Purpose	Parameter(s) Type	Result
abs(x)	<cmath>	Returns the absolute value of its argument: abs(-7) = 7	int (double)	int (double)
ceil(x)	<cmath>	Returns the smallest whole number that is not less than x: ceil(56.34) = 57.0	double	double
cos(x)	<cmath>	Returns the cosine of angle: x: cos(0.0) = 1.0	double (radians)	double
exp(x)	<cmath>	Returns e^x, where e = 2.718: exp(1.0) = 2.71828	double	double
fabs(x)	<cmath>	Returns the absolute value of its argument: fabs(-5.67) = 5.67	double	double
floor(x)	<cmath>	Returns the largest whole number that is not greater than x:floor(45.67) = 45.00	double	double
islower(x)	<cctype>	Returns 1 (true) if x is a lowercase letter; otherwise, it returns 0 (false); islower('h') is 1 (true)	int	int
isupper(x)	<cctype>	Returns 1 (true) if x is an uppercase letter; otherwise, it returns 0 (false); isupper('K') is 1 (true)	int	int
pow(x, y)	<cmath>	Returns x^y; if x is negative, y must be a whole number: pow(0.16, 0.5) = 0.4	double	double
sqrt(x)	<cmath>	Returns the nonnegative square root of x; x must be nonnegative: sqrt(4.0) = 2.0	double	double
tolower(x)	<cctype>	Returns the lowercase value of x if x is uppercase; otherwise, it returns x	int	int
toupper(x)	<cctype>	Returns the uppercase value of x if x is lowercase; otherwise, it returns x	int	int

2. CHARACTER FUNCTIONS:

Function	Description	Example
isalnum()	This function is used to check whether a character is an alphanumeric(i.e A to Z or a to z) or (0 to 9).	isalnum('a')=True(True means non zero value). isalnum('&')=False
isalpha()	This function is used to test whether the given character is alphabetic character or not.	isalpha('a')=True(1) isalpha('5')=False(0)
isdigit()	This function is used to check whether or not it is a digit(0 to 9).	isdigit('9')=True isdigit('a')=False
islower()	This function is used to check whether the given character is lower case or not(a to z).	islower('a')=True islower('A')=False
isupper()	This function is used to check whether the given character is uppercase letter or not(A to Z).	isupper('a')=False Isupper('A')=True

3. STRING FUNCTIONS:

Expression	Effect
strVar.at(index)	Returns the element at the position specified by index.
strVar[index]	Returns the element at the position specified by index.
strVar.append(n, ch)	Appends n copies of ch to strVar, in which ch is a char variable or a char constant.
strVar.append(str)	Appends str to strVar.
strVar.clear()	Deletes all the characters in strVar.
strVar.compare(str)	Returns 1 if strVar > str; returns 0 if strVar == str; returns -1 if strVar < str.
strVar.empty()	Returns true if strVar is empty; otherwise, it returns false.
strVar.erase()	Deletes all the characters in strVar.
strVar.erase(pos, n)	Deletes n characters from strVar starting at position pos.

4. CONSOLE I/O FUNCTIONS

i. **Getchar():**The getchar( ) function returns a single character from a standard input device (keyboard). It takes no parameter and the returned value is the input character.

ii. **Putchar():**The putchar( ) function takes one argument, which is the character to be sent to output device. It also returns this character as a result. The general form of the putchar( ) function is :
putchar(ch);
where ch is a variable of type character.

iii. **Gets() function:** The gets( ) function gets a string terminated by a newline character from the standard input stream stdin. The gets ( ) replaces the newline by a null character (\0).

iv. **getch( ) and getche( ) functions**: The general form of the getch( ) and getche( ) is

```
ch = getche( ) ;
ch1 = getch( ) ;
```

ch and ch1 are the variables of type character. They take no argument and require the conio.h header file. On execution, the cursor blinks, the user must type a character. The value of the character returned from getche ( ) is assigned to ch.

5. **USER DEFINED C++ FUNCTIONS:** A C++ function is a grouping of program statements in a single unit. The main ( ) function is the starting point for the execution of a program. The definition of main ( ) would look like as follows :

```
main ( ) {
// main program statements
}
```

6. In C ++, the main ( ) returns a value of type int to the operating system. The functions that have a return value should use the return statement for termination.
7. **FUNCTION PROTOTYPE**: The function prototype informs the compiler about the functions to be used in a program, the argument they take and the type of value they return. Functions which do not return any value are known as void functions.
8. **ARGUMENTS TO A FUNCTION**: Sometimes the calling function supplies some values to the called function. These are known as parameters. The variables which supply the values to a calling function are called actual parameters. The variable which receive the value from called statement are termed formal parameters.
9. **RETURN TYPE OF A FUNCTION**: The return type, which specifies the type of the value that the function returns, or void if no value is returned. In C++11, auto is a valid return type that instructs the compiler to infer the type from the return statement.
10. **LOCAL VARIABLE**: A variable declared within the body of a function will be evaluated only within the function. The portion of the program in which a variable is retained in memory is known as the scope of the variable. The scope of the local variable is a function where it is defined.
11. **GLOBAL VARIABLE**: A variable that is declared outside any function is known as a global variable. The scope of such a variable extends till the end of the program. These variables are available to all functions which follow their declaration. : : a represents the global variable. The symbol : : is called scope resolution operator.
12. **STORAGE CLASSES**: The storage class of a variable determines which parts of a program can access it and how long it stays in existence. The storage class can be classified as (i) Automatic (ii) Register (iii) Static (iv) External.

Storage Class	Keyword	Lifetime	Visibility	Initial Value
Automatic	auto	Function Block	Local	Garbage
External	extern	Whole Program	Global	Zero
Static	static	Whole Program	Local	Zero
Register	register	Function Block	Local	Garbage
Mutable	mutable	Class	Local	Garbage

13. **CALLING OF A FUNCTION**: The function can be called using either of the following methods:
    i. call-by-value
    ii. call-by-reference

	Call by value	Call by reference
1	A copy of value is passed to the function	An address of value is passed to the function
2	Changes made inside the function is not reflected on other functions	Changes made inside the function is reflected outside the function also
3	Actual and formal arguments will be created in different memory location	Actual and formal arguments will be created in same memory location

- **INLINE FUNCTIONS:** While the sequence of events may save memory space, it takes some extra time. To save execution time in short functions, inline function is used. Each time there is a function call, the actual code from the function is inserted instead of a jump to the function. The inline function is used only for shorter code.
- **FUNCTION WITH DEFAULT ARGUMENTS:** C ++ allows to call a function without specifying all its arguments. In such cases, the function assigns a default value to a parameter which does not have a matching argument in the function call. Default value are specified when the function is declared. The compiler knows from the prototype how many arguments a function uses for calling.
- **FUNCTION OVERLOADING:** The same function name can be used to create functions that perform a variety of different tasks. This is known as function polymorphism in OOP. The function would perform different operations depending on the argument list in function call. The correct function to be invoked is determined by checking the number and type of arguments.

CHECK YOURSELF

1. Which of the following statement is correct?
   A. Only one parameter of a function can be a default parameter.
   B. Minimum one parameter of a function must be a default parameter.
   C. All the parameters of a function can be default parameters.
   D. No parameter of a function can be default.

2. Unary scope resolution operator is denoted by

   A. ! !
   B. % %
   C. :
   D. : :

3. How many minimum numbers of functions need to be presented in c++?
   A. 0
   B. 1
   C. 2
   D. 3

4. The variable that are listed in the function's call are:
   A. Actual Parameters
   B. Declared Parameters
   C. Passed Parameters
   D. None of them

5. Which from the following is not a storage class specifier in C++?
   A. Auto
   B. Register
   C. Extern
   D. Mat

STRETCH YOURSELF

1. Explain the difference between Call by Value and Call by Reference with the help of an example?
2. What is Function Overloading?
3. What are the different storage class specifiers used in C++?
4. Write a program to describe any 5 string functions?
5. Write a program to describe any 5 in built mathematical function?

ANSWERS

Answers to Check Yourself:

1. C
2. D
3. B
4. A
5. D