Chapter 3

Functions

Concept of modular programming

Modular programming is the act of designing and writing programs as functions. A large program is broken down into sub-programs(Modules). The process of breaking large programs into smaller sub programs is called modularization. The subprograms are also called functions.
Functions are classified into two bulit-in or predefined functions and user defined functions. Built-in functions are readily available to the users and are stored in header files. They are a part of C++ Library. User defined functions are written by the user for specific tasks. Two methods are used for modular programming top-down and bottom-up method.

ഫങ്ഷനുകളുപയോഗിച്ച് പ്രോഗ്രാമുകളെ തയ്യാറാക്കുന്നതാണ് മോഡുലാർ പ്രോഗ്രാമിംഗ്. വലിയ പ്രോഗ്രാമിനെ ചെറിയ പ്രോഗ്രാമുകളാക്കി (മൊഡ്യൂൾ) മാറ്റുന്ന പ്രക്രിയയാണ് മോഡുലറൈസേഷൻ. ഇങ്ങനെ തയ്യാറാക്കുന്ന ചെറിയ പ്രോഗ്രാമുകളാണ് ഫങ്ഷൻ.
ഫങ്ഷനുകളെ bulit-in അല്ലെങ്കിൽ predefined functions എന്നും user defined functions എന്നും രണ്ടായി തിരിക്കാം. C++ ലൈബ്രറിയിലുള്ള ഹെഡർ ഫയലുകളിൽ സൂക്ഷിച്ചിട്ടുള്ളവയാണ് Built-in functions. പ്രത്യേക ആവശ്യങ്ങൾക്കായി ഒരാൾ തയ്യാറാക്കുന്നതാണ് User defined functions. Modular programming ൽ ഉപയോഗിക്കുന്ന രണ്ട് രൂപങ്ങളാണ് top-down and bottom-up.

Advantages of Modular programming

a)Reduction in program size.
Modular approach helps to isolate repeated task. Instructions are grouped together to form functions and is invoked by using its name. Thus program size is reduced.

പ്രോഗ്രാമിന്റെ വലിപ്പം കുറയ്ക്കുന്നു.
ആവർത്തിക്കുന്ന task കളെ വേർതിരിച്ച് പ്രത്യേകം പേരിലുള്ള ഫങ്ഷനുകളാക്കുമ്പോൾ പ്രോഗ്രാമിന്റെ വലിപ്പം കുറയുന്നു.

b)Less chance of error
When size of program is reduced syntax error will be less. Modularity helps to reduce logical error.

തെറ്റാനുള്ള സാധ്യത കുറയുന്നു.
പ്രോഗ്രാമിന്റെ വലിപ്പം കുറയുമ്പോൾ സിന്റാക്സ് ലോജിക്കൽ എററുകൾ കുറയുന്നു.

c)Reduce programming complexity
Modularity helps to reduce program complexity by dividing larger programs into functions.

പ്രോഗ്രാമിന്റെ സങ്കീർണ്ണത കുറയ്ക്കുന്നു.
വലിയ പ്രോഗ്രാമിനെ ഫങ്ഷനുകളാക്കുമ്പോൾ സങ്കീർണ്ണമായ പ്രോഗ്രാം ലഘുവായി മാറുന്നു.

d)Improve reusability
Once a function is written it can be used later. It helps to reduce development time.

പുനരുപയോഗ്യത കൂട്ടുന്നു.
ഒരു തവണ എഴുതുന്ന ഫങ്ഷൻ അവശ്യമുള്ളപ്പോൾ പിന്നീട് ഉപയോഗിക്കാവുന്നതാണ്.

Dis-advantages of Modular programming
Modular programming requires more time and memory.

മോഡുലാർ പ്രോഗ്രാമിംഗ് കൂടുതൽ സമയവും മെമ്മറിയും ഉപയോഗിക്കുന്നു.

Role of main() function

The *main()* is a user defined function in C++. The execution of a C++ program begins and ends at main(). By default the main() function returns an integer value to the Operating system.

C++ ലെ ഒരു user defined function ആണ് *main()*. C++ program ന്റെ പ്രവർത്തനം തുടങ്ങുന്നതും അവസാനിക്കുന്നതും main() ലാണ്. main() ഓപ്പറേറ്റിംഗ് സിസ്റ്റത്തിന് ഇന്റിജർ മടക്കിക്കൊടുക്കുന്നു.

Predefined functions

Console functions for character I/O

The console I/O functions are functions which reads directly from the keyboard and outputs to the monitor. These functions require the inclusion of header file cstdio in the program.
getchar( ) function: returns a single character from the keyboard.
Example:
char ch=getchar( )

കീബോർഡ് വഴി ഇൻപുട്ട് സ്വീകരിക്കുകയും മോനിട്ടറിൽ കാണിക്കുകയും ചെയ്യുന്നതാണ് console I/O functions. cstdio എന്ന ഫയൽ ഉൾപ്പെടുത്തിയാൻ മാത്രമേ ഇത്തരം ഫങ്ഷനുകൾ ഉപയോഗിക്കാൻ സാധിക്കൂ.
getchar( ) function: കീബോർഡിൽ നിന്നും ഒരു ക്യാരക്ടർ മാത്രമെടുക്കുന്നു.

putchar( ) function: displays or prints a single character.

putchar( ) function: ഒരു ക്യാരക്ടറിനെ മാത്രം

Example:
char ch='P';
putchar(ch); //Displays P on the screen.

കാണിക്കുന്നു.
Example:
char ch='P';
putchar(ch); //Displays P on the screen.

Stream functions for I/O operations

C++ supports a number of input/output operations. The stream acts as an intermediate between I/O devices and the user. A stream is a group of bytes. Streams are available in the form of functions stored in iostream header file.

യൂസറിനും ഇൻപുട്ട് ഔട്ട്പുട്ട് ഉപകരണങ്ങൾക്കുമിടയിൽ കാണുന്ന മാധ്യമമാണ് സ്ട്രീം. ഇത് ബൈറ്റുകളുടെ കൂട്ടമാണ്. സ്ട്രീമുകളെ ഉപയോഗിക്കാനുള്ള ഫങ്ഷനുകൾ iostream എന്ന ഹെഡർ ഫയലിലാണുള്ളത്.

Input functions

a) get( )

The get( ) function is accept a single character or stream through the keyboard.
ഒരു ക്യാരക്ടർ മാത്രം ഇൻപുട്ട് ചെയ്യാനുള്ള ഫങ്ഷനാണ് get( )

```
char ch;
ch=cin.get(ch); //accepts a character and stores in ch
cin.get(ch); //accepts a character and stores in ch.
```

b) getline( )

The getline( ) function is used to accept a string through the keyboard. It reads a string until a newline character is read.
The format of getline function is
cin.getline(line,size);
Where line is the name of the variable which stores line of text and size specifies the maximum number of characters which can be stored in line.

Example:

```
#include<iostream>
using namespace std;
int main( )
{
char name[40];
cout<<"Enter the name";
cin.getline(name,40);
cout<<name;
return 0;
}
```

Newline നൽകുന്നതുവരെയുള്ള ഒരുകൂട്ടം ക്യാരക്ടറുകൾ ഇൻപുട്ട് ചെയ്യാനുള്ള ഫങ്ഷനാണ് cin.getline(line,size); ഇവിടെ size എന്നത് പരമാവധി ഉൾപ്പെടുത്താവുന്ന ക്യാരക്ടറുകളുടെ എണ്ണമാണ്.

Output functions

The output functions allows flow of bytes(stream) from memory to output object cout

cout ഒബ്ജക്ട് ഉപയോഗിച്ച് stream കളെ ഔട്ട്പുട്ട് ചെയ്യാനുള്ള ഫങ്ഷനുകൾ

a)Put( ) function

The put( ) function is used to display a single character.

Example:
char ch='c';
cout.put(ch);

ഒരു ക്യാരക്ടർ മാത്രം ഔട്ട്പുട്ട് ചെയ്യാനുള്ള ഫങ്ഷനാണ് put( )

b)write( ) function

The write( ) function is used to displays a string.

Example:
char str[10]="hello";
cout.write(str,10);

String നെ കാണിക്കാനുള്ള ഫങ്ഷനാണ് write( )

String functions

A string is a group of characters. String functions are used to manipulate a string. The header file cstring is used for these functions.

string കളെ ഉപയോഗപ്പെടുത്തുവാനുള്ള ഫങ്ഷനുകൾ cstring എന്ന ഹെഡർ ഫയൽ വഴി ഉപയോഗിക്കാം

Function	Use	Syntax	Example

i) strlen( )	The strlen( ) function is used to find the length of a string(number of characters in the string).	int strlen(string);	n = strlen("Computer"); Stores 8 in n(ie number of charactes in the String Computer).
ii) strcpy( )	The strcpy( ) function is used to copy one string into another.	strcpy(string1,string2);	strcpy (st,"Think"); copies constnt string "Think" to st.

iii) strcat( )	The strcat function is used to join (concatenate) one string to another string. The length of the resultant string is the total length of the two strings.	strcat(string1,string2);	char s1[20] = "Welcome", s2[10] = " All";
			strcat(s1,s2);
			cout << s1;
			//Displays Welcome All.

iv) strcmp( )	The strcmp( ) function is used to compare two strings. The funtion returns, 0 if both strings are equal. -ve value if string1 is alphabetically lower than string2. +ve value if string1 is alphabetically higher than string2 .	strcmp(string1,string2)	strcmp("Ajay","sanju"); returns -ve. strcmp("small","big") returns +ve. strcmp("Ace","Ace") returns 0
v) strcmpi( )	The strcmpi( ) function is used to compare two strings ignoring cases. The function will treat both the upper case and lower case letters as same for comparison. The function returns, 0 if the two strings are identical(same). a negative value if string1<string2. a positive value if string1>string2.	strcmpi(string1,string2)	strcmpi("Ajay","sanju"); returns -ve. strcmpi("small","big") returns +ve. strcmpi("Ace","ACE") returns 0
Mathematical functions The header file cmath is to be used for mathematical functions.			
i) abs( )	It returns the absolute value of an integer.	abs(int)	abs(-25),Returns 25
ii) sqrt()	It returns the square root of a number.	sqrt(double)	sqrt(36),Returns 6.
Iii) pow( )	It is used to find power of a number.	pow(double,double);	Pow(2,3);Returns 8 (ie,2*2*2=8)
Character functions Character functions are used to perform various operations on characters. The header file cctype) supports these functions. The commonly used character functions in C++ are,			
i) isupper( )	This function checks if a character is in upper case or not. The function returns 1 if the character is in uppercase, and 0 otherwise.	int isupper(char)	isupper('A') --> 1 isupper('v') --> 0
ii) islower( )	This function checks if a character is in lower case or not. The function returns 1 if the character is in lowercase, and 0 otherwise.	int islower(char)	islower('X') --> 0 islower('b') --> 1
iii) isalpha( )	This function is used to check whether the given character is an alphabet or not. The function returns 1 if the character is an alphabet, and 0 otherwise.	int isalpha(char)	isalpha('q') --> 1 isalpha('G') --> 1 isalpha('1) --> 0 isalpha('$') --> 0
iv) isdigit( )	This function is used to check whether	int isdigit(char)	isdigit('q') --> 0

	the given character is a digit or not. The function returns 1 if the character is a digit, and 0 otherwise.		isdigit('G') --> 0 isdigit('1) --> 1 isdigit('$') --> 1
v) isalnum( )	This function is used to check whether a character is alphanumeric or not. The function returns 1 if the character is alphanumeric, and 0 otherwise.	int isalnum(char)	isalnum('q') --> 1 isalnum('G') --> 1 isalnum('1) --> 1 isalnum('$') --> 0
vi) toupper( )	This function is used to convert the given character into its uppercase.	char toupper(char)	touper('b') --> 'B'
vii) tolower( )	This function is used to convert the given character into its lower case.	char tolower(char)	tolower('Y') --> 'y'

User-defined functions

A user defined function is a group of code to perform a specific task. Every function in C++ function consists of two parts, A function header and a function body. A function definition consists of function header and function body.

പ്രത്യേക ജോലി ചെയ്യാൻ നിയോഗിച്ചിട്ടുള്ള പ്രോഗ്രാം കോഡുകളുടെ കൂട്ടമാണ് user defined function. C++ ഫങ്ഷന്റെ രണ്ട് ഭാഗമാണ് function header, function body എന്നിവ.

The general format of a function is

```
data_type function_name(argument_list) -->
{
statements ;
}
```

Header

body

The result of a function is called return value. A function can only return a value. A function which is defined cannot be executed by itself. A set of values passed to a function is called arguments. Body contains statements of the function. A function is invoked by another function such as main( ). Depending on type of communication, between calling and called function , functions are of four types.

ഫങ്ഷൻ മടക്കിനൽകുന്ന മൂല്യമാണ് return value. ഒരു ഫങ്ഷന് ഒരു return value മാത്രമേ സാധ്യമാകൂ. ഫങ്ഷന് നൽകുന്ന മൂല്യങ്ങളാണ് arguments. ഫങ്ഷൻ ചെയ്യേണ്ട നിർദ്ദേശങ്ങളാണ് body. main() അല്ലെങ്കിൽ മറ്റൊരു ഫങ്ഷനിൽക്കൂടി മാത്രമേ ഒരു ഫങ്ഷനെ ഉപയോഗിക്കാൻ സാധിക്കൂ. ഫങ്ഷനുകൾ നാല് വിധത്തിൽ കാണുന്നു.

Function with no argument and no return value.
Argument ഉള്ളതും return value ഇല്ലാത്തതുമായ ഫങ്ഷൻ
	void sum1( )
	{
	int a, b, s;
	cout<<"Enter 2 numbers: ";
	cin>>a>>b;
	s=a+b;
	cout<<"Sum="<<s;
	}

Function with argument and no return value.
Argument, return value എന്നിവ ഇല്ലാത്ത ഫങ്ഷൻ
	void sum3(int a, int b)
	{
	int s;
	s=a+b;
	cout<<"Sum="<<s;
	}

Function with argument and return value.
Argument, return value എന്നിവ ഉള്ള ഫങ്ഷൻ
	int sum4(int a, int b)
	{
	int s;
	s=a+b;
	return s;
	}

Function with no argument but with return value.
Argument ഇല്ലാത്തതും return value ഉള്ളതുമായ ഫങ്ഷൻ
	int sum2()
	{
	int a, b, s;
	cout<<"Enter 2 numbers: ";
	cin>>a>>b;
	s=a+b;

```
return s;
}
```

The return statement returns a value to the calling function and transfers the program control back to the calling function.

return സ്റ്റേറ്റ്മെന്റ് പ്രോഗ്രാം കണ്ട്രോൾ പ്രധാനപ്രോഗ്രാമിലേയ്ക്ക് മാറ്റി നൽകുകയും വാല്യൂ മടക്കി നൽകുകയും ചെയ്യുന്നു.

Function prototype

Function prototype models the actual function. It specifies the compiler the type and number of arguments a function use.

യഥാർത്ഥ ഫങ്ഷന്റെ മാതൃകയാണ് prototype. ഒരു ഫങ്ഷൻ ഉപയോഗിക്കുന്ന argument കളുടെ തരവും എണ്ണവും കമ്പയിലറിന് മനസിലാകാനുപയോഗിക്കുന്നു.

Actual and Formal parameters

The values which are passed into a function is called arguments or parameters. The arguments or parameters present in function definition is called formal parameter and those present in function call is called actual parameter.

ഫങ്ഷനിലേയ്ക്ക് കടത്തിവിടുന്ന വാല്യൂകളാണ് arguments അല്ലെങ്കിൽ parameters. function definition ൽ ഉപയോഗിക്കുന്നത് formal parameter, function call ൽ ഉപയോഗിക്കുന്നത് actual parameter.

For example

```
int main( )
{
int x,y;
void output(int x,int y);
//Actual parameters are x and y
}
```

```
void output(int a,int b)
//Formal parameters are a and b
{
int c;
c=a+b;
cout<<"Sum ="<<c;
}
```

Note:The number and type of Actual and formal parameters must be the same.

Actual, formal parameter കളുടെ എണ്ണവും തരവും ഒരുപോലെയായിരിക്കണം.

Passing arguments to a function

Based on the method of passing arguments, function calling methods can be classified into two
1. Call by Value method .
2. Call by Reference method.

ആർഗ്യുമെന്റുകൾ നൽകുന്ന രീതിയനുസരിച്ച് function calling രണ്ട് വിധത്തിലുണ്ട്.
1. Call by Value method .
2. Call by Reference method.

Call by value (Pass by value) method

This is the default argument passing method in C++. In this method, the value contained in the actual argument is passed to the formal argument ie, a copy of the actual argument is passed to the function. Any changes made to the formal argument does not effect actual argument.

actual argument ന്റെ പകർപ്പ് formal argument ലേയ്ക്ക് നൽകുകയാണിവിടെ ചെയ്യുന്നത്. formal argument ൽ വരുത്തുന്ന ഒരു മാറ്റവും actual argument നെ ബാധിക്കുന്നില്ല.

Example: Swapping two values using call by value

```
#include<iostream>
using namespace std;
void swap(int,int);
//function prototype declaration
int main( )
{
int a=10,b=50;
swap(a,b);//function call by value
cout<<"The values of a and b are :";
cout<<a<<" "<<b;
}
```

```
void swap(int x,int y)
//function definition
{
int t;
t=x;
x=y;
y=t;
cout<<"The values of x and y are :";
cout<<x<<" "<<y;
}
```

Working of the above program

```
t=10;
x=50;
y=10;
```

Call by reference (Pass by reference) method

In this method a reference of the actual argument is passed to the function. The actual argument and formal argument shares the same memory location. Here any changes made to formal argument effects or is reflected back to the actual arguments. The formal argument is preceded by '&' Operator.

ഇവിടെ actual argument ന്റെ അഡ്രസാണ് ഫങ്ഷന് നൽകുന്നത്. actual argument, formal argument എന്നിവ ഒരേ മെമ്മറി പങ്കിടുമ്പോൾ formal argument ൽ വരുന്ന ഓരോ മാറ്റവും actual argument ൽ കാണാൻ സാധിക്കും. formal argument നെ & ചിഹ്നമുപയോഗിച്ചാണ് കാണിക്കുന്നത്.

Example:Swapping two values using pass by reference

```
#include<iostream>
using namespace std;
int main( )
{
void swap(int &x,int &y);
//function prototype declaration
int a=10,b=50;
cout<<"The values of a and b before swapping:";
cout<<a<<" "<<b;
swap(a,b); //function call by value
cout<<"\n"<<"The values of a and b after swapping :";
cout<<a<<" "<<b;
}
```

```
void swap(int &x,int &y)
//function definition
{
int t;
t=x;
x=y;
y=t;
}
Before Swapping
After Swapping
t=10;
x=50;
y=10;
```

Difference between Call by value and Call by reference methods.

Call by value	*Call by reference*
Ordinary variables are used as formal parameters.	Reference variables are used as formal parameters.
Actual parameters can be a constant.	Actual parameters will be variables.
Changes made to formal arguments do not reflect actual arguments.	Changes made in the formal arguments do reflect in actual arguments.
Actual argument and formal arguments are stored in different memory locations.	Actual argument and formal arguments are stored in same memory locations.

Scope of variables and function

Scope of a variable is the portion of a program where the variable can be used(accessed).

ഒരു വേരിയബിൾ ഉപയോഗിക്കാൻ സാധിക്കുന്ന പ്രോഗാമിന്റെ ഭാഗമാണ് അതിന്റെ Scope.

The variables declared within the body of a block(Function) are called ***local variables*** and can be used within the block. The life of local variables ends with execution of last instruction of the function.

ഒരു ഫങ്ഷന്റെയുള്ളിൽ ഡിക്ലയർ ചെയ്തിരിക്കുന്നതും അതിനുള്ളിൽ മാത്രം ഉപയോഗിക്കാൻ കഴിയുന്നതുമായ വേരിയബിളാണ് local variable. ഫങ്ഷൻ പ്രവർത്തിച്ചുതീരുന്നതോടെ ഇവയുടെ ഗീവകാലവും അവസാനിക്കുന്നു.

The variables declared outside any function and which are accessible to all functions are called ***global variables***. It has life time through out the program execution.

എല്ലാ ഫങ്ഷനുകളുടെയും വെളിയിൽ ഡിക്ലയർ ചെയ്തിട്ടുള്ള വേരിയബിളാണ് global variable. പ്രോഗ്രാം അവസാനിക്കുന്നതുവരെ ഇതിന് ജീവകാലമുണ്ട്.

A function which is declared inside the body of another function is called a ***local function***.

മറ്റൊരു ഫങ്ഷന്റെയുള്ളിൽ ഡിക്ലയർ ചെയ്തിരിക്കുന്ന ഫങ്ഷനാണ് local function.

A function declared outside the function body of any other function is called a ***global function***. It can be used throughout the program. The scope of a global function is the entire program and that of a local function is only within the function where it is declared.

എല്ലാ ഫങ്ഷനുകളുടെയും വെളിയിൽ ഡിക്ലയർ ചെയ്തിരിക്കുന്ന ഫങ്ഷനാണ് global function. ഇതിനെ പ്രോഗ്രാമിലെവടെയും ഉപയോഗിക്കാൻ സാധിക്കും. global function ന് പ്രോഗ്രാം മുഴുവനും സാധ്യതയുള്ളപ്പോൾ local function ന് അത് ഡിക്ലയർ ചെയ്തയിടത്ത് മാത്രമേ സാധ്യതയുള്ളൂ.