OBJECT: To understand functions in C++ along with function declaration, definition and calling of function. To understand function return type and parameters.

FUNCTIONS

A function is a group of statements that together perform a task. Every C++ program has at least one function, which is main(), and all the most trivial programs can define additional functions.

You can divide up your code into separate functions. How you divide up your code among different functions is up to you, but logically the division usually is such that each function performs a specific task. Function provides following benefits:

- Provides reusability as we can use a function multiple times
- Make code organized and increase readability of a program as when the code increases is always difficult to read.
- Decrease the length of code
- If there is a bug in the code it can be detected easily when we divided the code in function according to tasks, thus provides maintainability

1. Declaring a function or Prototype:

A function declaration tells the compiler about a function's name, return type, and parameters. The general structure of a function declaration is as follows:

```
return_type function_name( Parameter List );
```

2. Defining a function:

A function definition provides the actual body of the function. The general form of a C++ function definition is as follows:

```
return_type function_name( parameter list ) {
   body of the function
}
```

A function definition consists of a function header and a function body.

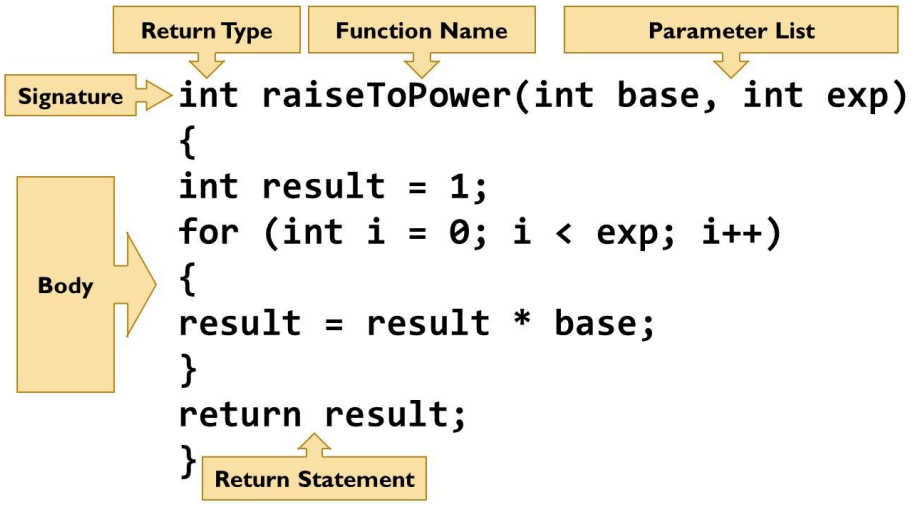
- ⇨ Return Type: A function may return a value. The return_type is the data type of the value the function returns. Some functions perform the desired operations without returning a value. In this case, the return_type is the keyword void.
- ⇨ Function Name: This is the actual name of the function.
- ⇨ Signature: The function name and the parameter list together constitute the function signature.
- ⇨ Parameters: A parameter is like a placeholder. When a function is invoked, you pass a value to the parameter. This value is referred to as actual parameter or argument. The parameter list refers to the type, order, and number of the parameters of a function. Parameters are optional; that is, a function may contain no parameters.
- ⇨ Function Body: The function body contains a collection of statements that define what the function does.

3. Calling a function:

To use a function, you will have to call or invoke that function. When a program calls a function, program control is transferred to the called function. A called function performs defined task and when it's return statement is executed or when its function-ending closing brace is reached, it returns program control back to the main program.

To call a function, you simply need to pass the required parameters along with function name, and if function returns a value, then you can store returned value. For example:

```
return_type = function_name(arguments);
```




```
int raiseToPower(int base, int exp)
{
int result = 1;
for (int i = 0; i < exp; i++)
{
result = result * base;
}
return result;
}
```


```
int main() {
int threeExpFour = raiseToPower(3, 4);
cout << "3^4 is " << threeExpFour << endl;
return 0;
}
```

PROGRAM 1: Demonstrates a simple function

```
#include <iostream>
using namespace std;

void starline(); //function declaration (prototype)

int main()
{
starline(); //call to function
cout << "Data type Range" << endl;
starline(); //call to function
cout << "char -128 to 127" << endl
<< "short -32,768 to 32,767" << endl
<< "int System dependent" << endl
<< "long -2,147,483,648 to 2,147,483,647" << endl;
starline(); //call to function
return 0;
}

// function definition
void starline() //function declarator
{
for(int j=0; j<45; j++) //function body
cout << '*';
cout << endl;
}
```

PROGRAM 2: Demonstrates a function which takes arguments

```
#include <iostream>
using namespace std;
void repchar(char, int); //function declaration
int main()
{
repchar('-', 43); //call to function
cout << "Data type Range" << endl;
repchar('=', 23); //call to function
cout << "char -128 to 127" << endl
<< "short -32,768 to 32,767" << endl
<< "int System dependent" << endl
<< "double -2,147,483,648 to 2,147,483,647" << endl;
repchar('-', 43); //call to function
return 0;
}
void repchar(char ch, int n) //function declarator
{
for(int j=0; j<n; j++) //function body
cout << ch;
cout << endl;
}
```

PROGRAM 3: Passing variable and expression to a function. Demonstrates function definition preceding function calls eliminating declaration.

```
#include <iostream>
using namespace std;

void repchar(char ch, int n) {
for(int j=0; j<n; j++) //function body
cout << ch;
cout << endl;
}

int main() {
for (int a = 180; a<=190 ; a++)
repchar(a, a-179);
return 0;
}
```

PROGRAM 4: Demonstrate a function which return value.

```
#include <iostream>
using namespace std;

float lbstokg(float);

int main() {
float lbs, kgs;
cout << "\nEnter your weight in pounds: ";
cin >> lbs;
kgs = lbstokg(lbs);
cout << "Your weight in kilograms is " << kgs << endl;
return 0;
}
float lbstokg(float pounds) {    // converts pounds to kilograms
float kilograms =  0.453592 * pounds;
return kilograms;
}
```

PROGRAM 5: Demonstrate a function preceding main which return value and eliminates unnecessary variables.

```
#include <iostream>
using namespace std;

float lbstokg(float pounds) {
return 0.453592 * pounds;
}

int main() {
float lbs;
cout << "\nEnter your weight in pounds: ";
cin >> lbs;
cout << "Your weight in kilograms is " << lbstokg(lbs)<< endl;
return 0;
}
```

PROGRAM 6: Demonstrate that return statement can come anywhere in function.

```
#include <iostream>
using namespace std;
void printNumberIfEven(int num)
{
if (num % 2 == 1) {
return;
}
cout << "even number; number is " << num << endl;
}

int main() {
int x;
cout<<"Enter a number:";
cin>>x;
printNumberIfEven(x);
return 0;
}
```

PROGRAM 7: Demonstrate power function.

```
#include <iostream>
#include <string>
using namespace std;
long raiseToPower(int base, int exp) {
int result = 1;
for (int i = 0; i < exp; i++) {
result = result * base;
}
return result;
}

int main() {
int b,ex;
cout<<"Enter Base:";
cin>>b;
cout<<"Enter Power:";
cin>>ex;
cout << b << " ^ " << ex << " = "
      << raiseToPower(b,ex);
return 0;
}
```

PASSING ARRAYS TO FUNCTIONS

Arrays can be used as arguments to functions. In a function declaration, array arguments are represented by the data type and size of the array. For Example:

```
void somefunc( int elem[5] );
void display(char[3][3]);
```

You can also write declaration without size:

```
void somefunc( int elem[] );
```

Function doesn't need the size of the first dimension as two dimensional array is an array of arrays. It doesn't need to know how many elements there are, but it does need to know how big each element is. Far Example:

```
void display(char[][3]);
```

PROGRAM 8: Demonstrates a double dimensional array by creating a board for tic-tac-toe by using a function which is taking array as an argument.

```
#include<iostream>
#include<conio.h>
using namespace std;

void display(char[3][3]);

int main(){
char arr[3][3]={{' ',' ',' '},
                {' ',' ',' '},
                {' ',' ',' '}};
for (int i=0 ; i<3 ; i++)
   for (int j=0 ; j<3 ; j++)
   {
   display(arr);
   cout<<"Please Enter X or O";
   cin>>arr[i][j];
   system("cls");
   }
}
void display(char a[3][3])
{
cout<<"\n\n\n  -----------\n";
for (int i=0 ; i<3 ; i++){
   for (int j=0 ; j<3 ; j++)
   cout<<" | "<<a[i][j];
cout<<" |\n  -----------\n\n";
}
}
```

Exercise 1:

Write a program which contain a user define function to convert the temperature from Fahrenheit to degree Celsius. The function should take one argument and return one value. Take the value of Fahrenheit from user and the program should give the value in Celsius.

Use the formula: C = (F - 32) × 5/9

```
Enter Temperature in Fahrenheit: 100
Temperature in Celsius is: 37.7778
```

Exercise 2:

Write a function called hms_to_secs() that takes three int values—for hours, minutes, and seconds—as arguments, and returns the equivalent time in seconds (type long). Create a program that uses this function by obtaining a time value in hours, minutes, and seconds from the user (format 24:59:59), calling the function, and displaying the value of seconds it returns. If user enters hour value greater than 24 or minuets or second value greater than 60 the program should again ask for the values.

```
Enter Hours: 2
Enter Min: 23
Enter Sec: 12
The Value in Seconds is: 8592
Exiting Program
```

```
Enter Hours: 12
Enter Min: 70

Wrong Value Renter

Enter Hours: 1
Enter Min: 20
Enter Sec: 44
The Value in Seconds is: 4844
Exiting Program
```

Exercise 3:

Write a function called reversit() that reverses a C-string (an array of char). Use a for loop that swaps the first and last characters, then the second and next-to-last characters, and so on. The string should be passed to reversit() as an argument. Write a program to exercise reversit(). The program should get a string from the user, call reversit(), and print out the result. Use an input method that allows embedded blanks. Test the program with Napoleon's famous phrase, "Able was I ere I saw Elba."

```
Enter a String: welcome all
Reversed: lla emoclew
```