# Functions & Recursion

A function is an individual set of instructions invoke able for a single purpose. Functions allow to structure programs in segments of code to perform individual tasks. In C++ a function is a group of statements that is given a name, and which can be called from some point of the program. Following codes will explain the idea.

les_07_code_01.cpp

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

int square(int num1)
    {
    int squared;
    squared = num1 * num1;
    return squared;
    }

int main()
    {
    int my_num;
    cout<<"Enter a number : ";
    cin>>my_num;
    cout<<"Square of "<<my_num<<" is "<<square(my_num);
    return 0;
    }
```

les_07_code_02.cpp

Alternate implementation of les_07_code_01.cpp

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

int square(int num1)
     {
     return num1*num1;
     }

int main()
      {
      int my_num;
      cout<<"Enter a number : ";
      cin>>my_num;
      cout<<"Square of "<<my_num<<" is "<<square(my_num);
      return 0;
      }
```

les_07_code_03.cpp

Function with multiple parameters

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

int maximum(int a, int b, int c)
   {
   int largest;
   if (a>b)
   largest = a;
   else
   largest = b;
   if (c>largest)
   largest = c;
   return largest;
   }

int main()
   {
   int num1, num2, num3, max;
   cout<<"Enter three numbers : ";
   cin>>num1>>num2>>num3;
   max = maximum(num1,num2,num3);
   cout<<"Largest number you entered is "<<max;
   return 0;

   }
```

les_07_code_04.cpp

Function call from other then main

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

double ftoc(double temp)
{
return (temp-32.0)*(5.0/9.0);
}

double ctof(double temp)
{
return (temp*(9.0/5.0) + 32.0);
}

double convertTemp(double temp, char scale)
{
if(scale == 'c' || scale == 'C')
{
cout<<"Converted from F to C"<<endl;
return ftoc(temp);
}

else if (scale == 'f' || scale == 'F')
{
cout<<"Converted from C to F"<<endl;
return ctof(temp);
}
else
{
cout<<"*****ERROR*****"<<endl<<"Invalid Scale"<<endl;
return 0;
}
}

int main()
{
double act_temp, conv_temp;
char conv_to;
cout<<"Enter temperature and unit to convert to : ";
cin>>act_temp>>conv_to;
conv_temp=convertTemp(act_temp,conv_to);
cout<<"The converted temperature is "<<conv_temp<<endl;
return 0;
}
```

les_07_code_05.cpp

Predicate Functions

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

bool isEven(int num)
{
if (num%2 == 0)
return true;
else
return false;
}

int main(void)
{
int val;
cout<<"Enter a number to test : ";
cin>>val;
if(isEven(val))
cout<<val<<" is Even";
else
cout<<val<<" is Odd";
return 0;

}
```

les_07_code_06.cpp

Predicate function to check vowel

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

bool isVowel(char letter)
{
if(letter=='a'||letter=='e'||letter=='i'||letter=='o'||letter=='u')
return true;
else
return false;
}

int main()
{
char ltr;
cout<<"Enter an alphabet in lower case only : ";
cin>>ltr;
if(isVowel(ltr))
cout<<ltr<<" is a vowel.";
else
cout<<ltr<<" is a consonant.";
return 0;
}
```

les_07_code_07.cpp

Void Functions

```
#include<iostream>
using namespace std;
void Heading()
{
cout<<"************************************"<<endl;
cout<<"** EE-163 Computers & Programming **"<<endl;
cout<<"**      FE - Electrical           **"<<endl;
cout<<"************************************"<<endl;
}

int main(void)
{
```

```
Heading();
return 0;
}
```

les_07_code_08.cpp

Void Functions

```
#include<iostream>
using namespace std;
void Heading(string course, string batch)
{
cout<<"*************************************"<<endl;
cout<<"** "<<course<<" **"<<endl;
cout<<"**      "<<batch<<"           **"<<endl;
cout<<"*************************************"<<endl;
}

int main(void)
{
Heading("EE-163 Computers & Programming","FE - Electrical");
return 0;
}
```

les_07_code_09.cpp

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

int main()
{
int num1, num2;
num1 = 13;
num2 = 12;
cout<<num1<<endl;
cout<<num2<<endl;
int temp;
temp = num2;
num2 = num1;
num1 = temp;
cout<<num1<<endl;
cout<<num2<<endl;
return 0;
}
```

les_07_code_10.cpp

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

void swap(int a, int b)
```

```
{
int temp;
temp = b;
b = a;
a = temp;

}

int main()
{
int num1, num2;
num1 = 13;
num2 = 12;
cout<<num1<<endl;
cout<<num2<<endl;
swap(num1, num2);
cout<<num1<<endl;
cout<<num2<<endl;
return 0;
}
```

les_07_code_11.cpp

Pass by reference

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

void swap(int &a, int &b)
{
int temp;
temp = b;
b = a;
a = temp;

}

int main()
{
int num1, num2;
num1 = 13;
num2 = 12;
cout<<num1<<endl;
cout<<num2<<endl;
swap(num1, num2);
cout<<num1<<endl;
cout<<num2<<endl;
return 0;
}
```

## Recursion

les_07_code_12.cpp

infinite_recursion.cpp

```
#include<iostream>
using namespace std;
void infinite_recursion (void);

//main
int main ()
{
cout<<"Making an infinite call";
infinite_recursion();
}

void infinite_recursion (void)
{
cout<<endl<<"Function call within function";
infinite_recursion();
}
```

les_07_code_13.cpp

factorial with recursion

```
#include<iostream>

using namespace std;

unsigned long factorial(unsigned long val); //Prototype for factorial function
int main ()
{
unsigned long num;
cout<<endl<<"Enter a No. to find its factorial ";
cin>>num;

cout<<endl<<num<<" ! is = "<<factorial(num);

return 0;

}

unsigned long factorial(unsigned long val)
{
if(val == 1 || val == 0)
{
return 1;
}

if(val>1)
{
cout<<endl<<"At Recursive call val was :"<<val;
return val* factorial(val-1);
}
}
```

les_07_code_14.cpp

exponent of a number with recursion

```
#include<iostream>

using namespace std;
// Prototype
int intpower(int base, int exp);
// Calculates power for int base and exponent
int main(void)
{
int a,b;
cout<<"\nEnter two integers:";
cin>>a>>b;
cout<<"\n\n"<<a<<" ^ "<<b<<" = "<<intpower(a,b);
return 0;
}
// Definition
int intpower(int base,int exp)
{
if(base==0)  // Base case 1
{
return 0;
}
if(exp==0)   // Base case 2
{
return 1;
}
if(exp==1)   // Base case 3
{
return base;
}

if(exp>1)   // Inductive step
{
return base*intpower(base,exp-1);
}
}
```

les_07_code_15.cpp

Multiple Functions

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

int fibonacci(int num);   // Function Prototype for fibonacci()
int main(void)
{
int a,b;
cout<<endl<<"Enter the Fibonacci number you wish to find:\n";
cin>>a;
b=fibonacci(a);           // Function Call
cout<<"\nTerm "<<a<<" of Fibonacci sequence is "<<b<<endl;
return 0;
}
int fibonacci(int num)        // Function Prototype
{
if(num==0)
{
return 0;
}
if(num==1)
{
return 1;
}
if(num>1)
{
return fibonacci(num-1)+fibonacci(num-2);
}
}
```