|  |
| --- |
| **NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES**  **CS 201–DATA STRUCTURES LAB**  **Lab Session 04** |
| **Instructors:** Mr.Irfan, Ms. Mubashra |

Outline

* Functions
* Recursion
* Direct and Indirect Recursion
* Task

**FUNCTION**

A function is a group of statements that together perform a particular task. Every C++ program has at least one function and that is a main function.

Based on the nature of task, we can divide up our program into several functions.

***What are local variables?***

*These are the variables that are declared in the function. Their lifetime ends when the execution of the function finishes and are only known in the function in which they are declared.*

## Function returns a value

Value returning functions are used when only one result is returned and that result is used directly in an expression.

***General Format of a function return a value***

***datatype nameOfFunction()***

***{***

***return variable;***

***}***

## Void Function

Void functions are used when function doesn’t return a value.

***General Format of a void function***

***void nameOfFunction()***

***{***

***Statement1; Statement2;***

***…***

***Statement n;***

***}***

**RECURSION**

# When a function repeatedly calls itself, it is called a recursive function and the process is called recursion.

It seems like a never ending loop, or more formally it seems like our function will never finish. In some cases, this might true, but in practice we can check if a certain condition becomes true then return from the function.

***Base Case***

***The case/condition in which we end our recursion is called a base case.***

***Example of finite recursion***

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

***void myFunction( int counter)***

***{***

***if(counter == 0)***

***return;***

***else***

***{***

***cout <<counter<<endl; myFunction(--counter); return;***

***}***

***}***

***int main()***

***{***

***myFunction(10);***

***}***

## Characteristics of Recursion

Every recursion should have the following characteristics.

1. A simple base case which we have a solution for and a return value. Sometimes there are more than one base cases.
2. A way of getting our problem closer to the base case. i.e. a way to chop out part of the problem to get a somewhat simpler problem.
3. A recursive call which passes the simpler problem back into the function.

***General Format***

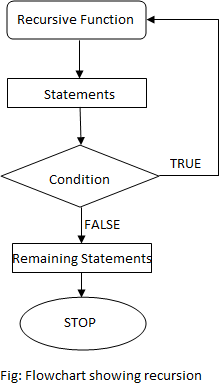
returntype recursive\_func ([argument list])

{

statements;

recursive\_func ([actual argument])

}



**DIRECT Vs INDIRECT RECURSION**

There are two types of recursion, direct recursion and indirect recursion.

## Direct Recursion

A function when it calls itself directly is known as Direct Recursion.

***Example of Direct Recursion***

#include<iostream> using namespace std; int factorial (int n)

{

if (n==1 || n==0) return 1;

else

return n\*factorial(n-1);

}

int main()

{

int f = factorial(5); cout << f;

}

## Indirect Recursion

A function is said to be indirect recursive if it calls another function and the new function calls the first calling function again.

***Example of Indirect Recursion***

#include<iostream> using namespace std; int func1(int);

int func2(int); int func1(int n)

{

if (n<=1) return 1;

else

return func2(n);

}

int func2(int n)

{

return func1(n-1);

}

int main()

{

int f = func1(5); cout << f;

}

Here, recursion takes place in 2 steps, unlike direct recursion.

* First, func1 calls func2
* Then, func2 calls back the first calling function func1.

## Disadvantages of Recursion

* Recursive programs are generally slower than non recursive programs. This is because, recursive function needs to store the previous function call addresses for the correct program jump to take place.
* Requires more memory to hold intermediate states. It is because, recursive program requires the allocation of a new stack frame and each state needs to be placed into the stack frame, unlike non-recursive(iterative) programs.

**Exercise:**

***Question #1***

Write a recursive function named Sum\_Num(int\* , int) which receives an integer array and it’s size and returns the sum of even numbers in the array. Call this function from main. Use appropriate parameters and return type.

***Question #2***

Write a recursive function to reverse the array.

***HomeTask:***

Write a recursive function for binary search.