Problem Solving and Programming 2

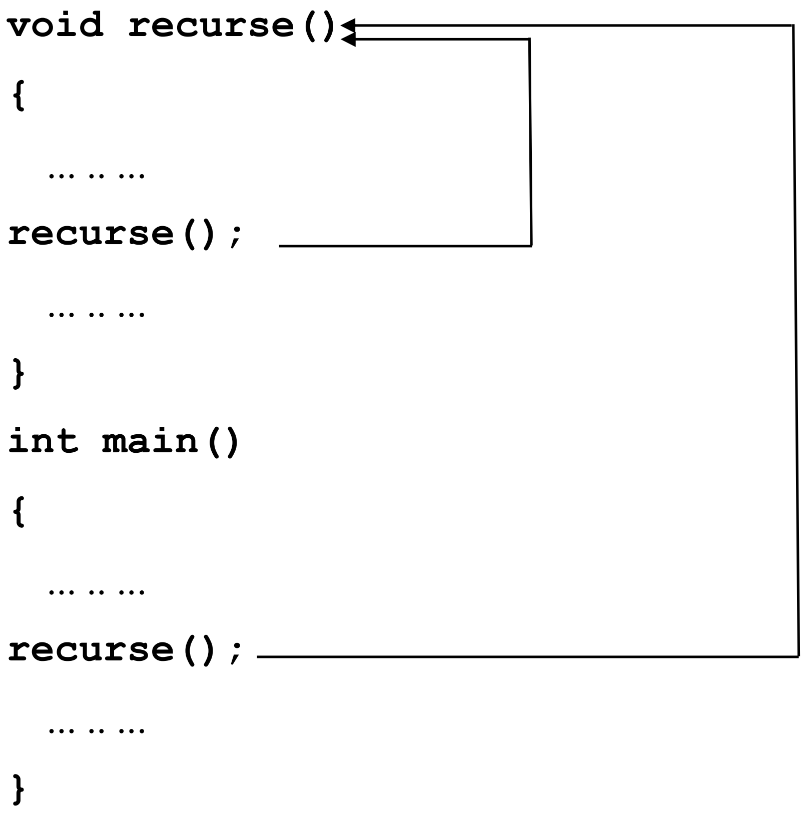
Lecture Two

Recursion in C++

- A function can call another function and sometimes may call itself.
- A function that calls itself is known as **recursive function**. And, this technique is known as **recursion**.
- Recursive function typically divides the problem into two conceptual pieces: a piece that the function knows how to do and a piece that it does not know how to do.
- It is terminated when the main condition no longer continues to be satisfied.

How does recursion work in C++?

- Recursion, takes the following general style:



```
void recurse()
{
  … .. …
recurse();
  … .. …
}
int main()
{
  … .. …
recurse();
  … .. …
}
```

- The recursion continues until the termination condition is met.
- To prevent infinite recursion, if...else statement (or similar approach) can be used where one branch makes the recursive call and other doesn't.
- Therefore, it can be concluded that recursion consists of two key parts to work as follows:
  - **The recursion part:** which calls the recursion function
  - **The termination condition:** which stops the recursion function upon satisfying a certain condition
- As mentioned above, the problem is divided into number of smaller problems
- Each of these new problems look like the original, so the function calls a copy of itself to work on the smaller problem– this is referred to as a **recursive call** and is also called the **recursion step.**
- The **recursion step** often includes the keyword **return**, because its result will be combined with the portion of the problem the function knew how to solve to form the result passed back to the original caller, possibly main.
- Example 1: Factorial of a Number Using Recursion

```
// Factorial of n = 1*2*3*...*n
int factorial(int n)
{
    if (n > 1)
    {
        return n*factorial(n-1);
    }
    else
    {
        return 1;
    }
}
int main() {
int n;
cout<<"Enter a number to find factorial: ‟;
cin>>n;
cout<<"Factorial of " << n <<" = " << factorial(n);
 }
```

Output

```
Enter a number to find factorial: 4

Factorial of 4 = 24
```

Explanation: How this example works

- Suppose the user entered 4, which is passed to the factorial() function.

4 * factorial(3)
3 * factorial(2)
2 * factorial(1)
1

1- In the first **`factorial`**() function, test expression inside if statement is true. The return **`num*factorial(num-1);`**

statement is executed, which calls the second **`factorial`**() function and argument passed is num-1which is 3.

2- In the second **`factorial()`** function, test expression inside if statement is true. The return **`num*factorial(num-1)`**; statement is executed, which calls the third **`factorial`**() function and argument passed is num-1 which is 2.

3- In the third **`factorial`**() function, test expression inside if statement is true. The return **`num*factorial(num-1)`**; statement is executed, which calls the fourth **`factorial`**() function and argument passed is num-1 which is 1.

4- In the fourth **`factorial()`** function, test expression inside if statement is false. The **`return 1`**; statement is executed, which returns 1 to third factorial() function.

5- The third **`factorial`**() function returns 2 to the second **`factorial`**() function.

6- The second **`factorial`**() function returns 6 to the first factorial() function.

7- Finally, the first **`factorial`**() function returns 24 to the main() function, which is displayed on the screen.

Example: count down recursive function

```
void count_down(int n)

{

    if(n!=0)

    {

      cout<<n<<endl;

      n--;

    count_down(n);
    }
    else
    {
        cout<<n<<endl;
    }
}
int main()
{
    int k = 5;
    count_down(k);
}
```

Example: calculating the sum of all the numbers from n to m recursively:

```
int CalcSum(int n, int m)
 {
    int sum = n;
    if (n < m)
     {
       n++;
       return sum += CalcSum (n, m);
     }
  return sum;
}
int main()
```

```
{
cout<<"Enter number n: ";
int n, m;
cin>>n;
cout<<"Enter number m: ";
cin>>m;
int sum = CalcSum (n, m);
cout<< sum;
 }
```

Example: Check if a string can be read from both sides or not

```
bool check(string str)
 {
 if (str.length() <=1)
   return true;
 if (str[0] == str[str.length() - 1])
   {
     str = str.substr(1, str.length() - 2);
     return check(str);
    }
   else
   {  return  false;  }
 }
int main()
 {
 cout<<"Enter a string ";
 string str;
cin>>str;
if (check(str))
 {
   cout<<"Yes";
}
 else
   {
    cout<<"No";
   }
```

Exercises

1. Write a complete program in C++ that includes a recursive method called **`count_up()`.** The recursive method receives an integer number and counts from 0 to that number.
2. Write a complete program in C++ that includes a recursive method called **`power()`.** The recursive method receives two integer numbers *X* and *Y*. The method calculates X^Y . Note: Any number to power 0 equals 1.
3. Write a complete program in C++ that includes a recursive method called **`sum()`.** The recursive method receives an integer number N and returns the summation of numbers from 1 to *N*.
4. Write a complete program in C++ that includes a recursive method called **`count_decimal_digits()`.** The recursive method receives a double number *N* and returns the of decimal digits
5. Write a complete program in C++ that includes a recursive method called **`to_binary`**(). The recursive method receives an integer number N and prints the binary representation of *N*.