Lecture 5
Recursion
Objectives

- Understand the concept of recursion
- Design simple recursive algorithms and implement them in C language
- Understand the execution of a recursive module through the use of a trace
Topics

1. Definition
2. Basic Outline
3. Example: factorial of a number
4. Codification in C language
5. Features
6. Exercises
7. Information sources
Definition

- A module is recursive when in the sequence of statements that make it up, there is a direct or indirect call to itself.

- Many mathematical functions are naturally defined in a recursive way. For instance:
  - Factorial of a number n: The factorial of a number n is n multiplied by the factorial of n-1:
    \[
    \text{factorial} \ (n) = n \times \text{factorial} \ (n-1)
    \]
  - Power of x raised to n is x multiplied by the power of x raised to n-1:
    \[
    x^n = x \times x^{n-1}
    \]
Topics

1. Definition

2. **Basic Outline**

3. Example: factorial of a number

4. Codification in C language

5. Features

6. Exercises

7. Information sources
Basic outline of a recursive module

- A recursive module always has:
  - **One or more base cases**
    - They don't include more recursive calls
    - They specify the *end condition* of the recursion
  - **One or more general (recursive) cases**
    - They include one or more recursive calls to itself
    - The recursive calls solve *smaller versions* of the initial task, so that there is a tendency towards the case base
Topics

1. Definition
2. Basic Outline
3. **Example: factorial of a number**
4. Codification in C language
5. Features
6. Exercises
7. Information sources
Example: Factorial of a number

\[
\text{factorial} \ (n) = n \times \text{factorial} \ (n-1)
\]

\[
\text{factorial} \ (3) = 3 \times \text{factorial} \ (2)
\]

\[
= 2 \times \text{factorial} \ (1)
\]

\[
= 1 \times \text{factorial} \ (0)
\]

\[
= 0 \times \ldots
\]

When does it finish?
Example: Factorial of a number

\[
\text{if } n \text{ is equal to 0 then} \quad \text{factorial} \ (n) = 1 \\
\text{else} \quad \text{factorial} \ (n) = n \ast \text{factorial} \ (n-1)
\]

The base case was missing!
Example: Factorial of a number

\[
\textbf{if} \ n \ \text{is equal to } 0 \ \textbf{then} \\
\text{factorial} \ (n) = 1 \\
\textbf{else} \\
\text{factorial} \ (n) = n \times \text{factorial} \ (n-1)
\]

Factorial (3) = 3 * factorial (2)

= 2 * factorial (1)

= 1

= 2

= 6
Topics

1. Definition
2. Basic Outline
3. Example: factorial of a number
4. **Codification in C language**
5. Features
6. Exercises
7. Information sources
Codification in C language

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

int factorial (int n)
{
    int res;
    if (n>0) // recursive case
        res = n * factorial(n-1);
    else    // base case
        res = 1;
    return res;
}

main()
{
    int num;
    cout << "Enter a number";
    cin >> num;
    cout << factorial(num);
}
```
Topics

1. Definition
2. Basic Outline
3. Example: factorial of a number
4. Codification in C language
5. Features
6. Exercises
7. Information sources
Features of recursion

- Recursion is suitable to solve problems that can naturally be defined in recursive terms
- A recursive solution always has an iterative equivalent solution
- A recursive solution needs a larger amount of memory to be executed
- A recursive solution is slower than an iterative one

```cpp
void write (int n){
    write (n/10);
    cout << n%10 << endl;
}
```
Trace of a recursion

```c
void base (int n){
    if (n<2) // base case
        cout << n;
    else{   // recursive case
        base (n/2);
        cout << n%2;
    }
}
```

```c
main()
{
    base(11);
}
```

```
Screen
1º  2º  3º  4º
```

```
base(11) -> base(5) -> base(2) -> base(1) = cout << 1
               
               
               
               
               
               
               
               
               
               
               
               
               
               
               
               
               
               
               
               
               
               
               
               
               
               
               
               
               
               
               
               
               
               
               
               
               
               
               
               
               
               
               
               
               
               
               
               
               
               
               
               
               
               
               
               } cout << 1
```
1. Which is the output on the screen if the call is `recursive(16)`?

   a) 00001
   b) 11111
   c) 10000
   d) 00100
   e) All the previous answers are incorrect

2. Which is the base case?
What does the following code do?

```c++
void reverse(char l){
    if (l == '.') // base case
        cout << endl;
    else{ // recursive case
        cin >> l;
        reverse(l);
        cout << l;
    }
}

main(){
    char letter;

    cout << "Enter a sentence finished by a full stop (dot) :");
    cin >> letter;
    reverse(letter);
    cout << letter;
}
```
Topics

1. Definition
2. Basic Outline
3. Example: factorial of a number
4. Codification in C language
5. Features
6. Exercises
7. Information sources
Exercises

1. Given a natural number \( n \), design a recursive module to show the growing sequence of natural numbers from 1 to \( n \), that is, 1 2 3 \( \ldots \) \( n \).

2. Given a natural number \( n \), design a recursive module to return the sum of the squared numbers from 1 to \( n \), that is, for \( n=4 \) the module must return 30 (\( 1^2+2^2+3^2+4^2=30 \)).

3. Given a natural number \( n \), design a module to show the digits of \( n \) in a reversed way. For instance, 2089 must be shown as 9802.

4. Design a recursive module that receives a number in decimal system and displays the same number in binary system. For instance, 12 in decimal system must be shown as 1100.

5. Given a natural number \( n \), design a recursive function to obtain the amount of odd digits in \( n \). For instance, \( \text{odd\_dig}(321)=2 \), \( \text{odd\_dig}(28)=0 \).
Topics

1. Definition
2. Basic Outline
3. Example: factorial of a number
4. Codification in C language
5. Features
6. Exercises
7. Information sources
Información sobre información:

- **Resolución de Problemas con C++**
  Walter Savitch
  - Capítulo 13 (Completo)

- **Fundamentos de Programación**
  Jesús Carretero, Félix García, y otros
  - Capítulo 7 (Apartado 7.5)

- **Problemas Resueltos de Programación en Lenguaje C**
  Félix García, Alejandro Calderón, y otros
  - Capítulo 5 (Apartado 5.3)