Lecture 1
Introduction
Tag cloud

BIT

BYTE

COMPILER

INTERPRETER

PROGRAM

ASCII

C LANGUAGE

ALGORITHM

C++
Objectives

• Understand differences between compilers and interpreters
• What is a program and which are the steps to develop computer programs?
• Some advice on how to learn to program
• Why is C our first programming language?
Topics

1. Representing the information
2. Compilers vs. Interpreters
3. What is a program?
4. How to develop a program?
5. How to learn to program?
6. The C language
7. Conclusions
8. Information sources
Representing information

- Computers represent information using two digits: **BINARY CODIFICATION** (base 2)

- **BIT** (BInary digIT): minimal unit of information that can be represented in a computer: 0 or 1

- **BYTE**: number of bits used to encode a single character of text: 8 bits

- **WORD**: natural unit of data used by a particular computer design: 1, 2, 3, 4 or 8 bytes (8, 16, 24, 32 or 64 bits)
Representing information

- A character is represented using a byte
- **Character encoding, character set or charset**: set of characters that can be codified in a computer. It is made up of:
  - Letters or alphabetic characters
  - Digits or numerical characters
  - Special characters and punctuation characters
  - Control characters (line break, …)
- **ASCII** character encoding system (*American Standard Code for Information Interchange*)
  - 01000001 → ‘A’
Topics

1. Representing the information
2. **Compilers vs. Interpreters**
3. What is a program?
4. How to develop a program?
5. How to learn to program?
6. The C language
7. Conclusions
8. Information sources
Compilers vs. Interpreters

Compiler

- A compiler **analyses** our program, checking its syntax and indicating the typographical errors, and it **generates** the program in machine language. It may need a **linkage** process when several library modules are joined.
Compilers vs. Interpreters

Interpreter

- The interpreter analyses and executes the program statement by statement
Topics

1. Representing the information
2. Compilers vs. Interpreters
3. **What is a program?**
4. How to develop a program?
5. How to learn to program?
6. The C language
7. Conclusions
8. Information sources
What is a program?

- A computer program is a set of ordered statements or instructions, written in a programming language for the computer to perform a specific task.
Topics

1. Representing the information
2. Compilers vs. Interpreters
3. What is a program?
4. How to develop a program?
5. How to learn to program?
6. The C language
7. Conclusions
8. Information sources
How to develop a program?

Understand the problem

Design a solution

Implement a program

Verify and debug the program
Understand the problem

Analyse the problem: answer the following question ....

What is the problem to be solved?

Identify

Input data → Process data → Output data
Understand the problem: example

Problem: calculate the mark of a subject

Calculate the mark of a subject for a student, considering the exam period (January or July).

In January, the final mark is obtained from the marks of the first practical exam (15%), the second practical exam (35%) and the written exam (50%), except when the written exam mark or the second practical exam mark is lower than 4. In this case, the final mark is the minimum of these marks.

In July, the final mark is obtained from the marks of the practical exam (50%) and the written exam (50%), except when one of these marks is lower than 4. In this case, the final mark is the minimum of these marks.
Design a solution

Propose the sequence of steps (algorithm) to solve the problem: answer the following question ....

How will the problem be solved?

Identify

Input data  
Algorithm  
Output data
Design a solution: example

**Algorithmic solution**

Algorithm:
- Enter the exam period to obtain the final mark
- If the exam is taken in January
  - Then
    - Enter the marks of the two practical exams and of the written exam
    - If the written exam mark < 4 or the 2nd practical exam mark < 4
      - Then
        YOUR FINAL MARK IS = Minimum (written ex. mark, 2nd pract. ex. mark)
      - else
        YOUR FINAL MARK IS = 0,15*Practice_1+0,35*Practice_2+0,5*Written
  - else
    YOUR FINAL MARK IS = 0,5*Practice+0,5*Written
- If the exam is taken in July
  - Then
    - Enter the written exam mark and the practice exam mark
    - If the written exam mark < 4 or the practice exam mark < 4
      - Then
        YOUR FINAL MARK IS = Minimum (written ex. mark, practice ex. mark)
      - else
        YOUR FINAL MARK IS = 0,5*Practice+0,5*Written
Implement a program

**Codify**, in a programming language, the sequence of steps to solve the problem

- You will have to
  1. know the **syntax** of a programming language
  2. **write** the program using a text editor
  3. **compile** and **correct** the syntax errors
# Implement a program

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

main()
{
    char call;
    float practice_1, practice_2, written_exam, final_mark;

cout << "Enter the exam call (J: January, L: July):";
cin >> call;

if (call == 'J') {
    cout << "Enter the mark of the practice exam 1:");
cin >> practice_1;
    cout << "Enter the mark of the practice exam 2:");
cin >> practice_2;
    cout << "Enter the mark of the written exam:");
cin >> written_exam;
    if (written_exam < 4 || practice_2 < 4)
        final_mark = minimum (written_exam, practice_2);
    else
        final_mark = 0.15*practice_1 + 0.35*practice_2 +
        0.5*written_exam;
}
if (call == 'L') {
    cout << "Enter the mark of the written exam :");
cin >> written_exam;
    cout << "Enter the mark of the practice exam:");
cin >> practice_2;
    if (written_exam < 4 || practice_2 < 4)
        final_mark = minimum (written_exam, practice_2);
    else
        final_mark = 0.5*practice_2 + 0.5*written_exam;
}
cout << "YOUR FINAL MARK IS = " << final_mark << endl;
}
```

- It may not compile
- Correct syntax errors

- It may fail when it is executed or when the input data are introduced
- Correct the logic errors

- It may not be the optimal solution
- Improve the temporal or spatial efficiency
Verify and debug the program

• Execute the program and correct the errors:
  1. Verify the program (Tests)
     – Execute the program and detect the errors
  2. Debug the program (Debugging)
     – Correct execution errors of the program

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
<th>OK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ja</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>Ja</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>Ja</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>Ja</td>
<td>5</td>
<td>9</td>
</tr>
<tr>
<td>P</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ju</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Ju</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>Ju</td>
<td>66</td>
<td>6</td>
</tr>
</tbody>
</table>
Topics

1. Representing the information
2. Compilers vs. Interpreters
3. What is a program?
4. How to develop a program?
5. How to learn to program?
6. The C language
7. Conclusions
8. Information sources
How to learn to program?

- Advice on how to learn to program
  - Study
  - Practice
  - Learn from errors
  - Search before asking
  - Share knowledge and experiences

- Programming is challenging, the best virtue of a programmer is perseverance
Topics

1. Representing the information
2. Compilers vs. Interpreters
3. What is a program?
4. How to develop a program?
5. How to learn to program?
6. The C language
7. Conclusions
8. Information sources
Why C language?

- A general purpose language
- It is one of the most widely used in business
- It favours a structured and modular programming
- We will use C language with some elements of C++ to make the learning process easier

C++ under the IMPERATIVE paradigm...

...not under its natural paradigm Object-Oriented, which will be taught in other subjects: Programming 2, Programming 3, ...
Creating an executable program in C

• You need

  • A text editor → to write the instructions.
    – Examples: Notebook, Kate, gedit
  
  • Compiler and linker → to generate the executable code
    – Example: g++

• The compilation process (source code → executable file) can be done by using:

  • Command line based programs: g++...
  
  • IDE (integrated development environment): Dev-C++, Eclipse, NetBeans...

IDEs include an editor, a compiler, a linker and a debugger, as well as other elements.
Topics

1. Representing the information
2. Compilers vs. Interpreters
3. What is a program?
4. How to develop a program?
5. How to learn to program?
6. The C language
7. Conclusions
8. Information sources
The importance of analysis and design

- **Understanding** the problem correctly is a **must** before thinking about the solution.

- Before writing the program (implementation) it is necessary to know **how to solve** the problem.
Topics

1. Representing the information
2. Compilers vs. Interpreters
3. What is a program?
4. How to develop a program?
5. How to learn to program?
6. The C language
7. Conclusions
8. Information sources
Recommended bibliography

- **Fundamentos de Programación**
  Jesús Carretero, Félix García, y otros
  ✔ Capítulo 1 (Apartados 1.5; 1.7)
  ✔ Capítulo 3 (Apartados 3.1; 3.2)

- **Problemas Resueltos de Programación en Lenguaje C**
  Félix García, Alejandro Calderón, y otros
  ✔ Capítulo 1 (Apartados 1.5; 1.7)

- **Resolución de Problemas con C++**
  Walter Savitch
  ✔ Capítulo 1