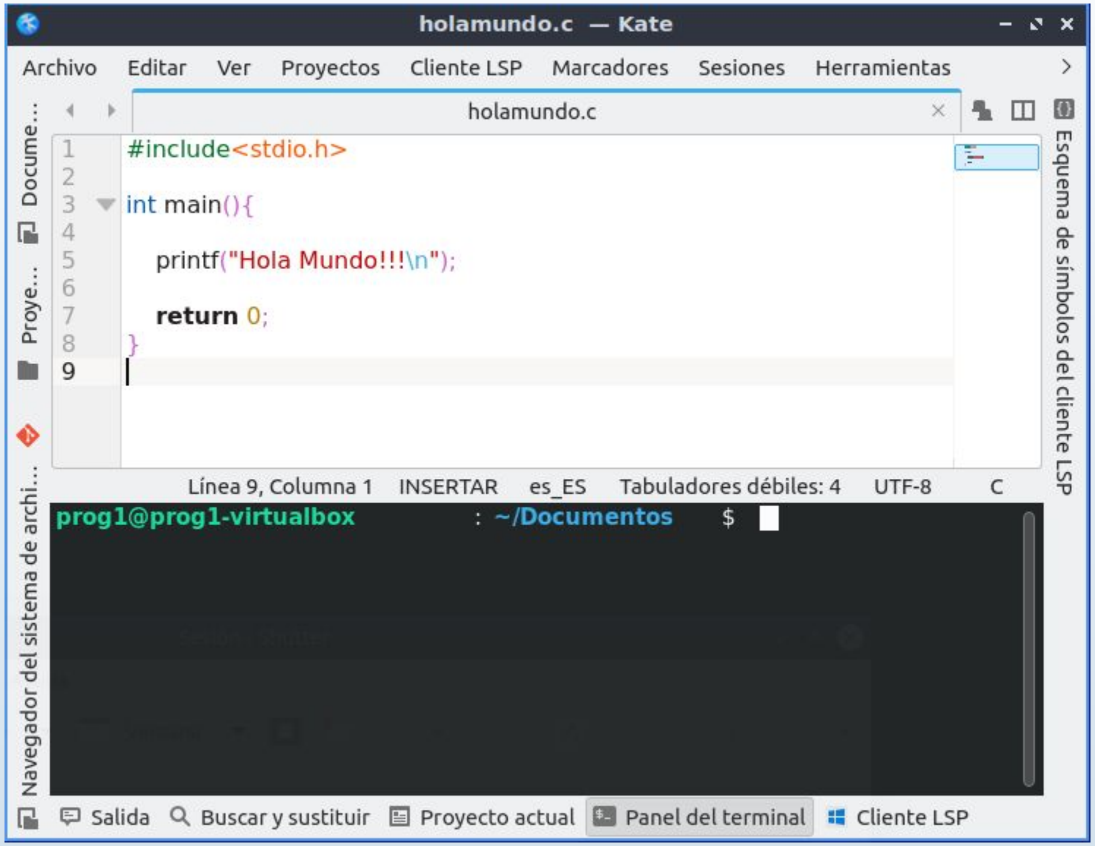


```c
#include<stdio.h>

int main(){

    printf("Hola Mundo!!!\n");

    return 0;
}

```

```
Línea 9, Columna 1   INSERTAR   es_ES   Tabuladores débiles: 4   UTF-8   C
```

```
prog1@prog1-virtualbox : ~/Documentos $
```

Programming 1

Lesson 1. Introduction

Degree in Computer Engineering

Index

1. Representation of information

- Computers represent information using two digits: BINARY CODING (base 2)
- BIT: (*BInary digIT*: 0 or 1) minimum representable unit of information in a computer.
- BYTE: 8 bits.
- *WORD*: The number of bits handled by a machine as a block. That is, it is the size (in bits) of the registers in the processor that have the capacity to store a word. The most commonly used sizes are 32 and 64 bits.

1. Representation of information

Representation of numbers

- Numbers are represented using a binary system

Decimal number	Representation in the binary system
0	00000000
1	00000001
15	00001111

- Negative numbers can be represented in several ways. One of the most common ways is the 2's complement representation.
- Real numbers can be represented in various ways. For example, in floating point notation (with mantissa and exponent).

1. Representation of information

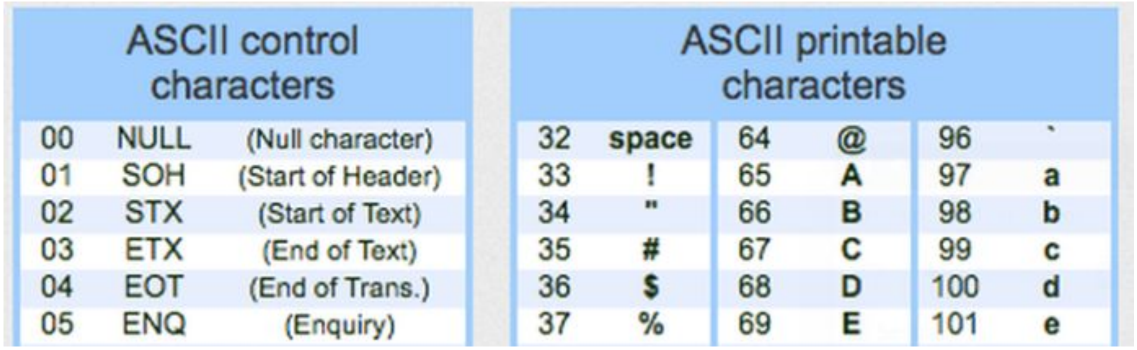
Representation of characters

- A character is represented by using one `byte`. The collection of characters that can be encoded in a computer is called a ***character set***, and is composed of:
  - alphabetic letters or characters
  - digits or numeric characters
  - special and punctuation characters
  - control characters (line breaks, etc.)

1. Representation of information

Character set

- **ASCII**. It allows to define 127 characters



ASCII control characters			ASCII printable characters					
00	NULL	(Null character)	32	space	64	@	96	`
01	SOH	(Start of Header)	33	!	65	A	97	a
02	STX	(Start of Text)	34	"	66	B	98	b
03	ETX	(End of Text)	35	#	67	C	99	c
04	EOT	(End of Trans.)	36	$	68	D	100	d
05	ENQ	(Enquiry)	37	%	69	E	101	e

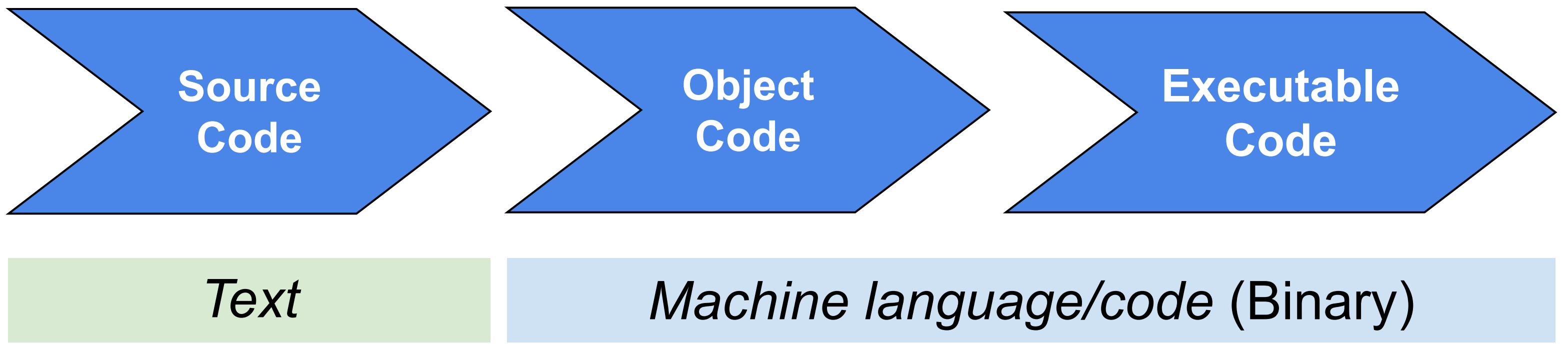
- **Extended ASCII**. it adds an extra bit and allows representation of accented vowels, *ñ*, etc..

- **Unicode**. Evolution of ASCII. It covers all the characters of all the world's orthographies.

2. Compilers vs. Interpreters

COMPILER

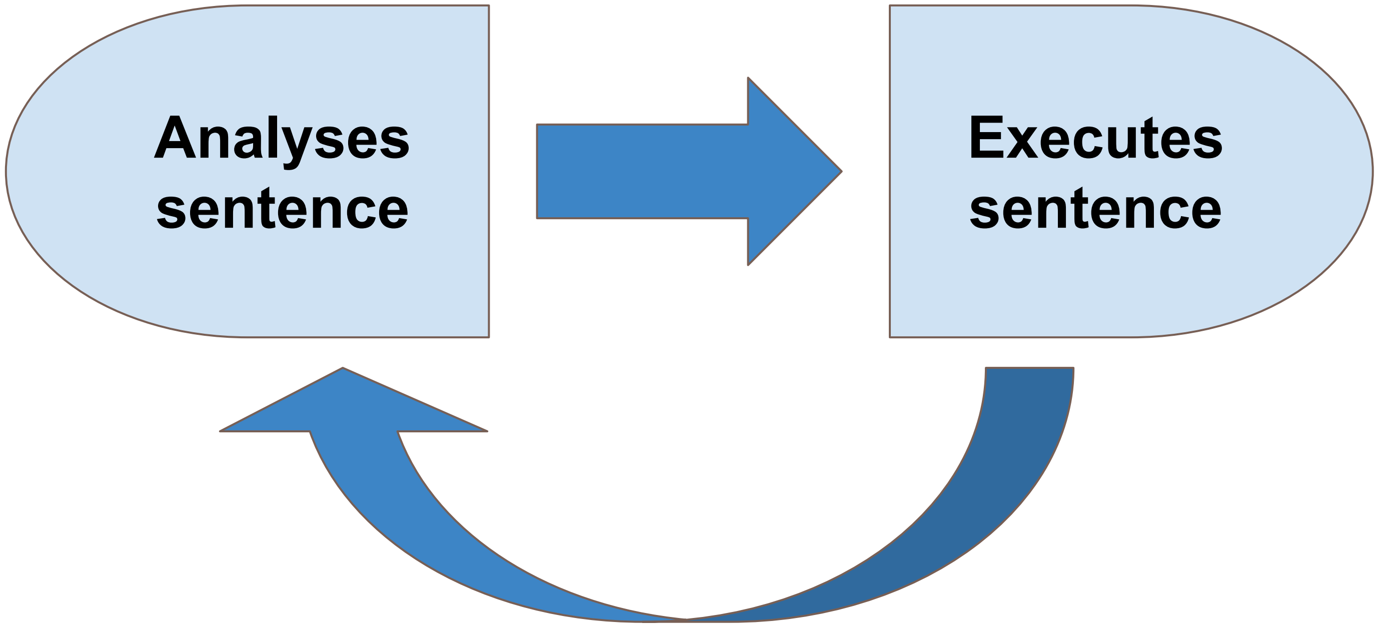
The compiler **analyses** your program, checking its syntax and indicating any typing errors, and **generates** the program in *machine language/code*. The program may require **linking**, where a number of library modules are attached to it.



2. Compilers vs. Interpreters

INTERPRETER

The interpreter analyses and executes the source code of a program in a sentence-by-sentence process.



3. Algorithm vs. computer program

ALGORITHM

Sequence of finite, well-defined steps that solve a problem.

Brushing your teeth

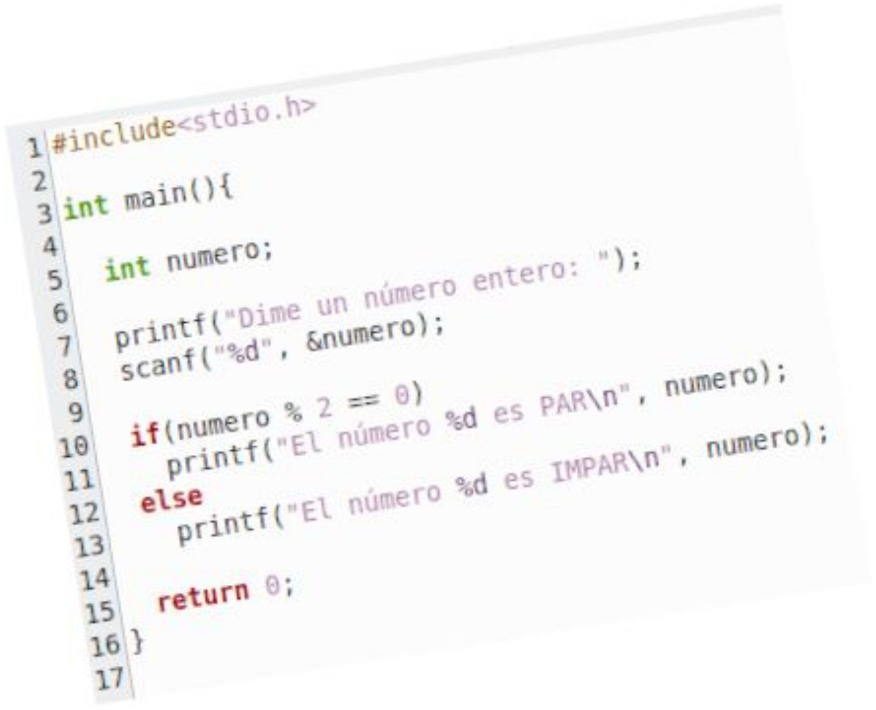
Making a Spanish omelette

Enrolling in the university

3. Algorithm vs. computer program

COMPUTER PROGRAM

A set of ordered instructions, written in a programming language, for a computer to carry out a given task.



```c
#include<stdio.h>

int main(){

   int numero;

   printf("Dime un número entero: ");
   scanf("%d", &numero);

   if(numero % 2 == 0)
      printf("El número %d es PAR\n", numero);
   else
      printf("El número %d es IMPAR\n", numero);

   return 0;
}
```

4. How to develop a computer program?

1. Understand the problem
- What needs to be solved?

2. Design a solution
- How will it be solved?

3. Write a computer program
- Code in a programming language

4. Test and debug the program
- Run the program and fix errors

4. How to develop a computer program?

Example:

Problem: Calculation of the final mark for a subject

In **January**, **15%** of the final mark comes from the mark obtained with the exercises done in class; **35%** comes from the mark of a computer-based exam; and the remaining **50%** comes from the mark of a written exam. It must be considered that if the mark of the written exam is lower than 4, the final mark will be the minimum between the calculated final mark and 4.5.

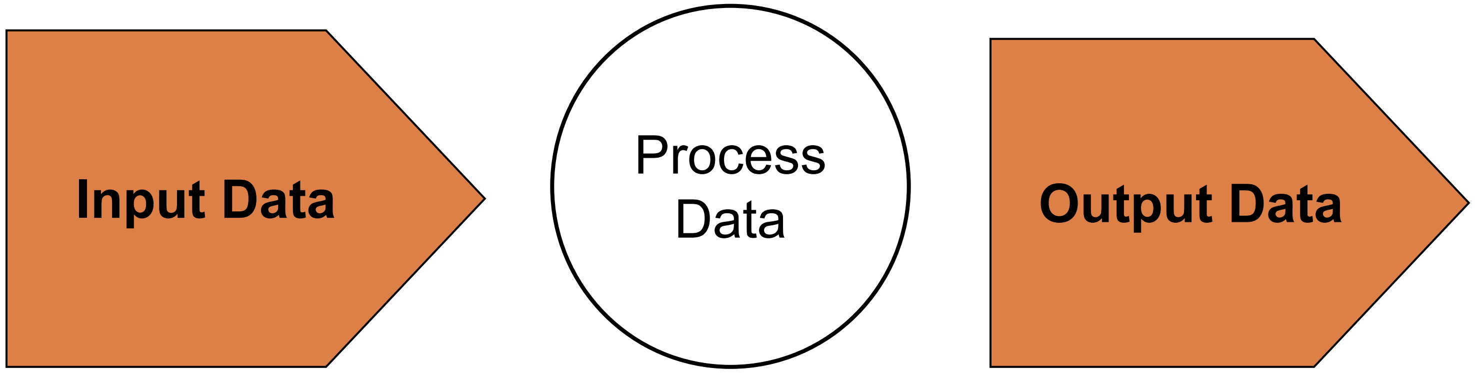
In **July**, **50%** of the final mark comes from a written exam, and the other **50%** comes from a computer-based exam. If any of these two marks is less than 4, the final mark will be the minimum between the calculated final mark and 4.5.

4. How to develop a computer program?

1. Understand the problem

Analyse the problem and answer the question:

WHAT needs to be solved?

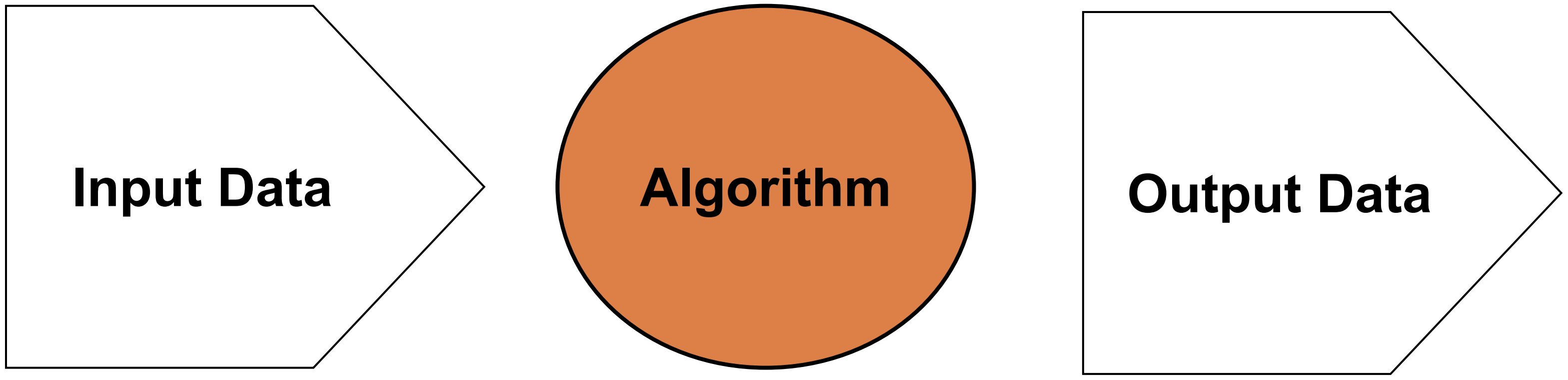


4. How to develop a computer program?

2. Design a solution

Propose the steps to be followed (**algorithm**) to solve the problem and answer the question:

HOW will it be solved?



4. How to develop a computer program?

2. Design a solution

Importance of analysis and design

- It is essential to **have a good understanding of the problem** before thinking about a solution.
- Before writing the program (coding), it is necessary to **have a clear idea on how to solve it**.

4. How to develop a computer program?

2. Design a solution

Algorithmic solution

Algorithm:

```
- Ask for which call you want to know the final mark.

- If (it is the January call)
  Then
  - Ask for the marks of the part of the classroom exercises, the computer-based exam
    and the written exam
  - If (the mark of the written exam < 4)
    Then
    - FINAL_MARK = minimum(0.15 * exercises + 0.35 * computer + 0.5 * written, 4.5)
    Else
    - FINAL_MARK = 0.15 * exercices + 0.35 * computer + 0.5 * written

- If (it is the July call)
  Then
  - Ask for the marks of the written exam and the computer-based exam.
  - If (the mark of the written exam < 4) o (the mark of the computer exam < 4)
    Then
    - FINAL_MARK = minimum(0.5 * computer + 0.5 * written)
    Else
    - FINAL_MARK = 0.5 * computer + 0.5 * written
```

4. How to develop a computer program?

3. Write a computer program

Code in a programming language the steps to be followed to solve the problem:

a. Know the syntax of the programming language to be used.
b. Write the program with a text editor.
c. Compile and fix syntactic errors.

4. How to develop a computer program?

```
#include <stdio.h>
int main() {
   char   call;
   float  exercices, computer, written, final_mark;

   printf("Choose the call (J - January, L - July): ");
   scanf("%s", &call);
   if (call == 'J') {
      printf("Write the mark of the exercises done in classroom: ");
      scanf("%f", &exercices);
      printf("Write the mark of the computer-based exam: ");
      scanf("%f", &computer);
      printf("Write the mark if the written exam: ");
      scanf("%f", &written);
      if (written < 4)
         final_mark = minimum(0.15 * exercices + 0.35 * computer + 0.5 * written, 4.5);
      else
         final_mark = 0.15 * exercices + 0.35 * computer + 0.5 * written;
   }
   else if (call == 'L' ) {
      printf("Write the mark of the written exam: ");
      scanf("%f", &written);
      printf("Write the mark of the computer-based exam: ");
      scanf("%f", &computer);
      if (written < 4 || computer < 4)
         final_mark = minimum(0.5 * computer + 0.5 * written, 4.5);
      else
         final_mark = 0.5 * computer + 0.5 * written;
   }
   printf("YOUR FINAL MARK IS: %f\n", final_mark);
   return 0;
}
```

minimum is a function that calculates the smallest number between 2 numbers.

C program to calculate the final mark

4. How to develop a computer program?

4. Test and debug the program

Run the program and fix errors :

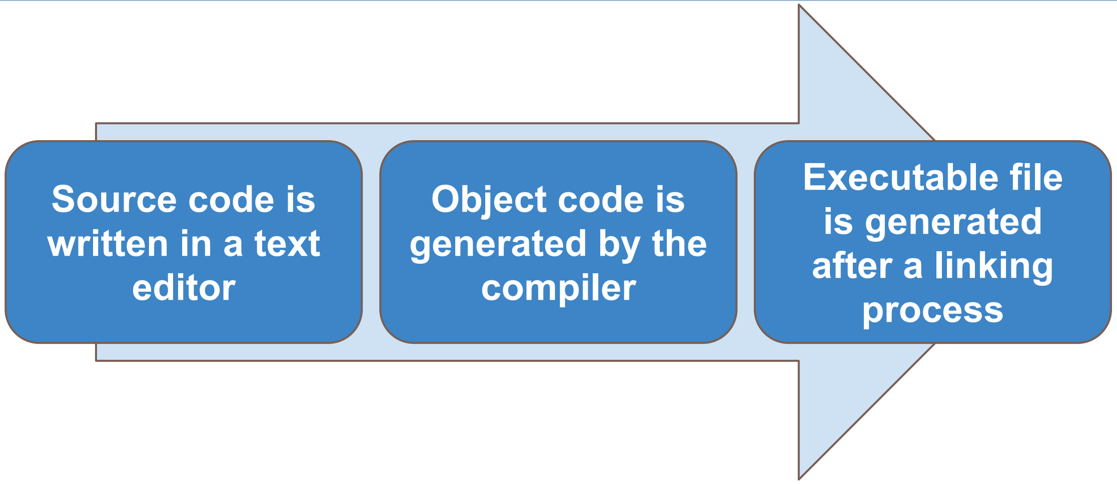
1. Test the program (**Tests**)
   - Run the program and **find errors**
2. Debug the program (**Debugging**)
   - **Fix execution errors** of the program

INPUT DATA				OUTPUT DATA	RESULT OK
Call	Exer	Comp	Written		
J	5	3	5	4.3	✓
J	3	6	5	5.05	✓
J	6	5	9	7.15	✓
P					✗
L		4	4	3.4	✓
L		10	5	6	✓
L		66	6	26.1	✗

5. Why do we use the C language?

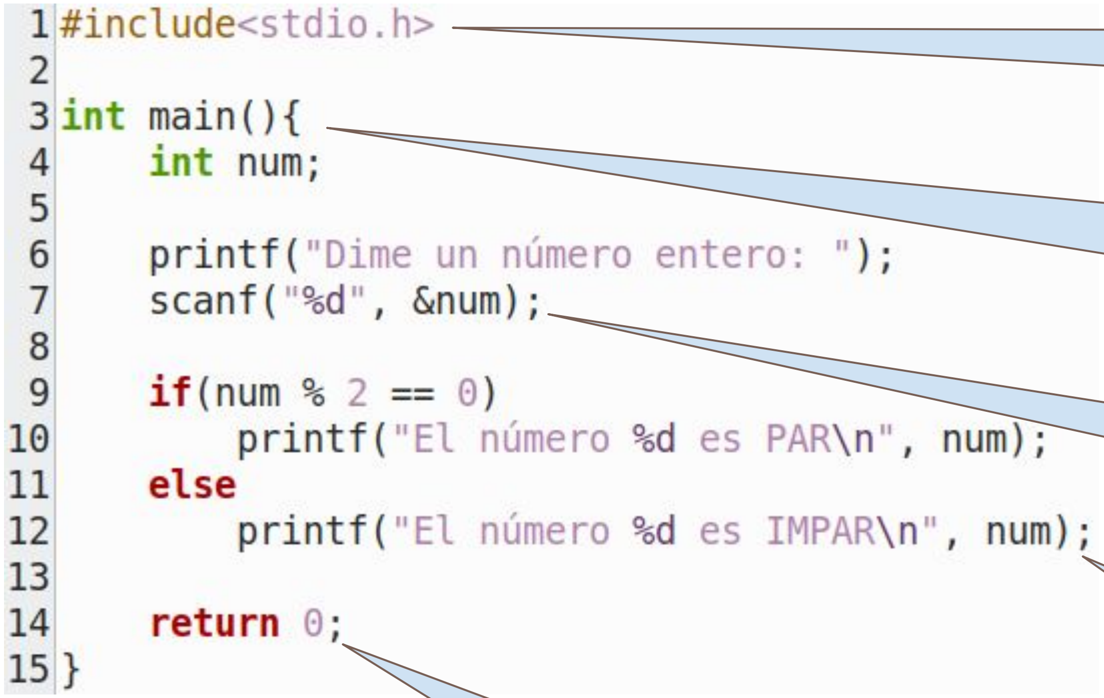
- It is a general purpose language
- Widely used in the job market
- Operating system independent
- Makes structured and modular programming easier
- Operates at low and high level
- Makes learning other programming languages easier

6. How to make executable programs


Source code is written in a text editor
Object code is generated by the compiler
Executable file is generated after a linking process

- The program **is written** in a text editor (kate, gedit, sublime text, etc.), giving rise to **the source code**.
- **It is compiled**, using the corresponding compiler, **to generate the executable file**. We will use **gcc** on the Linux operating system.
- Another possibility is to use an IDE (Integrated Development Environment). Example: Dev-C++ (Windows), Eclipse, NetBeans. IDEs include the editor, the compiler, the linker and a debugger, as well as other elements.

7. Structure of a C program



```c
#include<stdio.h>

int main(){
    int num;

    printf("Dime un número entero: ");
    scanf("%d", &num);

    if(num % 2 == 0)
        printf("El número %d es PAR\n", num);
    else
        printf("El número %d es IMPAR\n", num);

    return 0;
}
```

Inclusion of auxiliary files

`main` function. Main function of the program. It is the first function to be executed.

Read (input) sentence from keyboard

Write (output) statement to screen

Sentence to terminate the execution of the function and make it return a `0` value