Fundamentals of Programming I

1 Computers and Programming

Grado en Ingeniería Informática

Luis Hernández Yáñez
Facultad de Informática
Universidad Complutense

Index

Fundamentals of Programming I

Computer Science, Computers and Programming

Luis Hernández Yáñez

Computer Science and Computers

R.A.E.

Computer Science

Scientific knowledge and techniques that make the automatic processing of information possible by means of computers

Computer

Analog or digital electronic machine, with a large capacity memory and information processing methods, able to solve mathematic and logical problems with the execution of programs

Luis Hernández Yáñez

Computers

Everywhere and with multiple forms

Hardware and Software

Hardware

Components that make up the material part of a computer

Software

Programs, instructions and computer rules for executing tasks on a computer

Computer Programming

Programming

To tell a computer what it has to do

Program

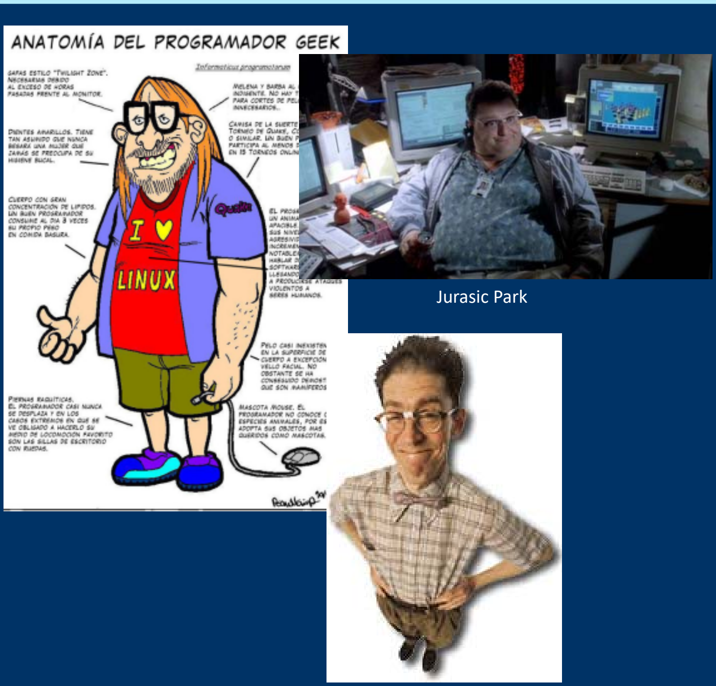
✓Sequence of instructions

✓Instructions the computer understands

✓To achieve a goal: *to solve a problem!*

Luis Hernández Yáñez

Programmers and Software Developers


ANATOMÍA DEL PROGRAMADOR GEEK
Informaticus programatorum
GAFAS ESTILO "TWILIGHT ZONE". NECESARIAS DEBIDO AL EXCESO DE HORAS PASADAS FRENTE AL MONITOR.
MELENA Y BARBA AL INDIGENTE. NO HAY T PARA CORTES DE PEL INNECESARIOS.
DIENTES AMARILLOS. TIENE TAN ASUMIDO QUE NUNCA BESARÁ UNA MUJER QUE JAMÁS SE PREOCUPA DE SU HIGIENE BUCAL.
CAMISA DE LA SUERTE TORNEO DE QUAKE, O SIMILAR. UN BUEN P PARTICIPA AL MENOS EN 15 TORNEOS ONLIN
CUERPO CON GRAN CONCENTRACIÓN DE LÍPIDOS. UN BUEN PROGRAMADOR CONSUME AL DÍA 3 VECES SU PROPIO PESO EN COMIDA BASURA.
Quake
I ♥ LINUX
EL PROG UN ANIMA APACIBLE SUS NIVE AGRESIVI INCREMEN NOTABLE HABLAR D SOFTWAR LLEGANDO A PRODUCIRSE ATAQUES VIOLENTOS A SERES HUMANOS.
PELO CASI INEXISTEN EN LA SUPERFICIE DE CUERPO A EXCEPCIÓN VELLO FACIAL. NO OBSTANTE SE HA CONSEGUIDO DEMOST QUE SON MAMÍFEROS
PIERNAS RAQUÍTICAS. EL PROGRAMADOR CASI NUNCA SE DESPLAZA Y EN LOS CASOS EXTREMOS EN QUE SE VE OBLIGADO A HACERLO SU MEDIO DE LOCOMOCIÓN FAVORITO SON LAS SILLAS DE ESCRITORIO CON RUEDAS.
MASCOTA MOUSE. EL PROGRAMADOR NO CONOCE ESPECIES ANIMALES, POR ES ADOPTA SUS OBJETOS MÁS QUERIDOS COMO MASCOTAS.

Jurasic Park

Teamwork
Multiple roles:

✓Project Engineer

✓Analysis Engineer

✓Design Engineer

✓Programmer

✓Tester

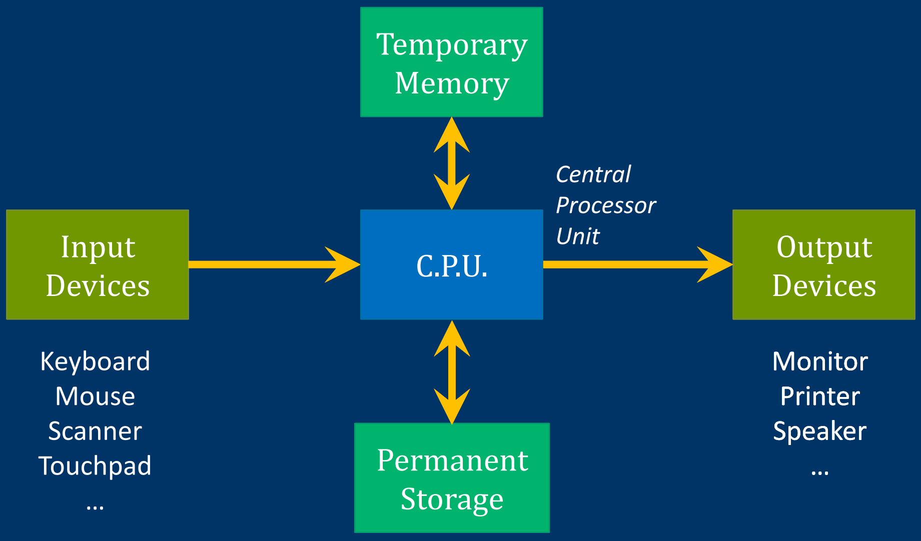
✓System Engineer

...

Luis Hernández Yáñez

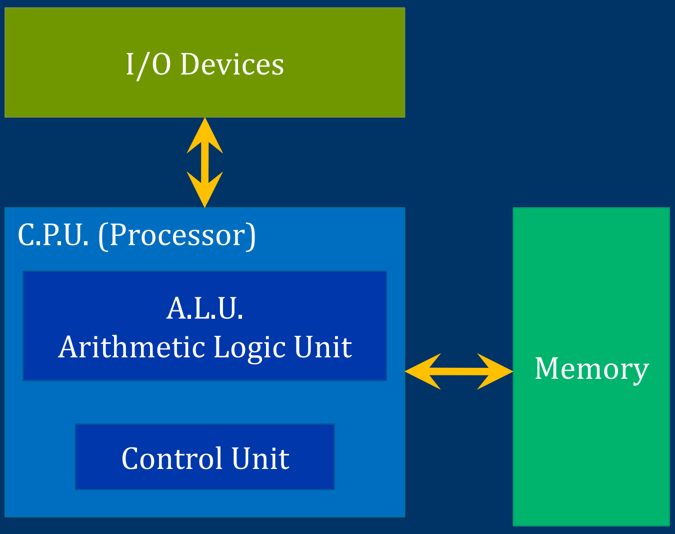
Computers

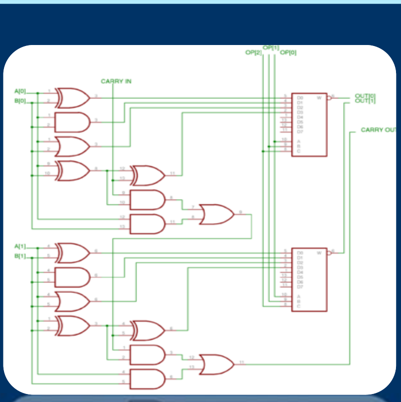
General scheme



Computers

Von Neumann Architecture

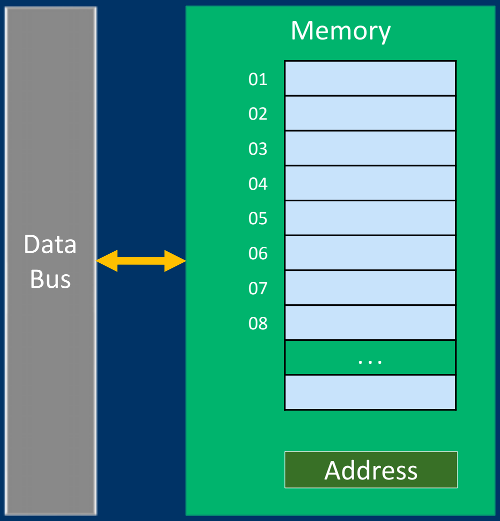




2-bit ALU (Wikipedia)

Computers

Memory



Data Bus

Memory

01
02
03
04
05
06
07
08
. . .

Address

Memory Cells (8/16/32/64 bits)

Each one at a Memory Address

Volatile

1 Bit = 0 / 1
1 Byte = 8 bits = 1 character
1 Kilobyte (Kb) = 1024 Bytes
1 Megabyte (Mb) = 1024 Kb
1 Gigabyte (Gb) = 1024 Mb
1 Terabyte (Tb) = 1024 Gb
1 Petabyte (Pb) = 1024 Tb

$2^{10} = 1024 \approx 1000$

Fundamentals of Programming I

Machine Language and Assembler

Computer Programming

Processors work with zeroes and ones (bits)

Basic Memory Unit: *Byte* (8 bits)
(2 hexadecimal digits: `01011011` → `0101 1011` → `5B`)

Machine Language

Hexadecimal codes for instructions, CPU registers, memory addresses or data

Instruction	*Meaning*
`A0 2F`	*Access memory cell with address 2F*
`3E 01`	*Copy cell in ALU Register 1*
`A0 30`	*Access memory cell with address 30*
`3E 02`	*Copy cell in ALU Register 2*
`1D`	*Add*
`B3 31`	*Save the result in memory cell with address 31*

Low-level language

Machine-dependent

Hard programming

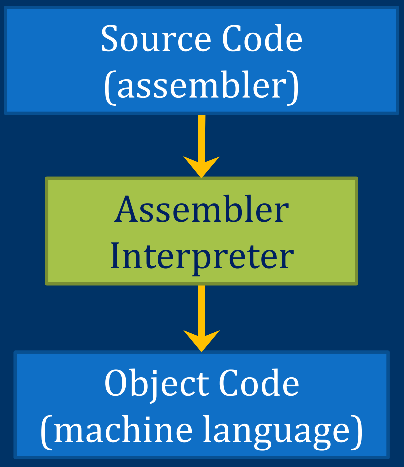
Assembler

Mnemonics for hexadecimal codes:

`A0` → `READ`   `3E` → `REG`   `1D` → `ADD`   …

Higher legibility:

```
READ 2F
REG 01
READ 30
REG 02
ADD
WRITE 31
```


Source Code
(assembler)

Assembler
Interpreter

Object Code
(machine language)

Middle-level language

Fundamentals of Programming I

High-Level Programming Languages

High-Level Programming Languages

- ✓ Closer to natural and mathematical languages

```
sum = operand1 + operand2;
```

- ✓ Higher legibility, much easier coding
- ✓ Data Structures / Procedural Abstraction

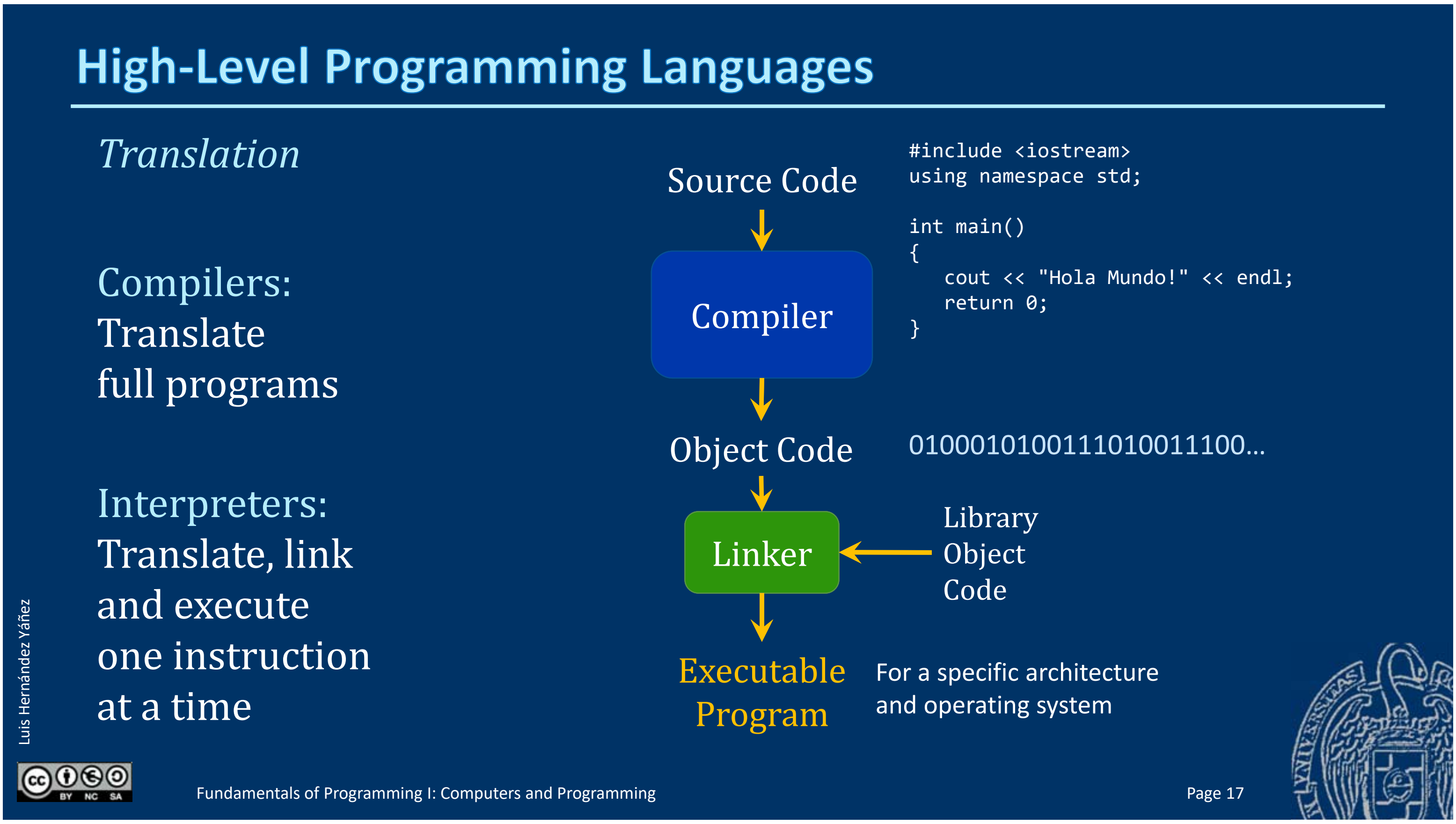
FORTRAN Python Prolog C#

C Pascal Cobol Lisp Ruby

BASIC Smalltalk Haskell Ada

Simula Java Eiffel C++

...



High-Level Programming Languages

Translation

Compilers:
Translate
full programs

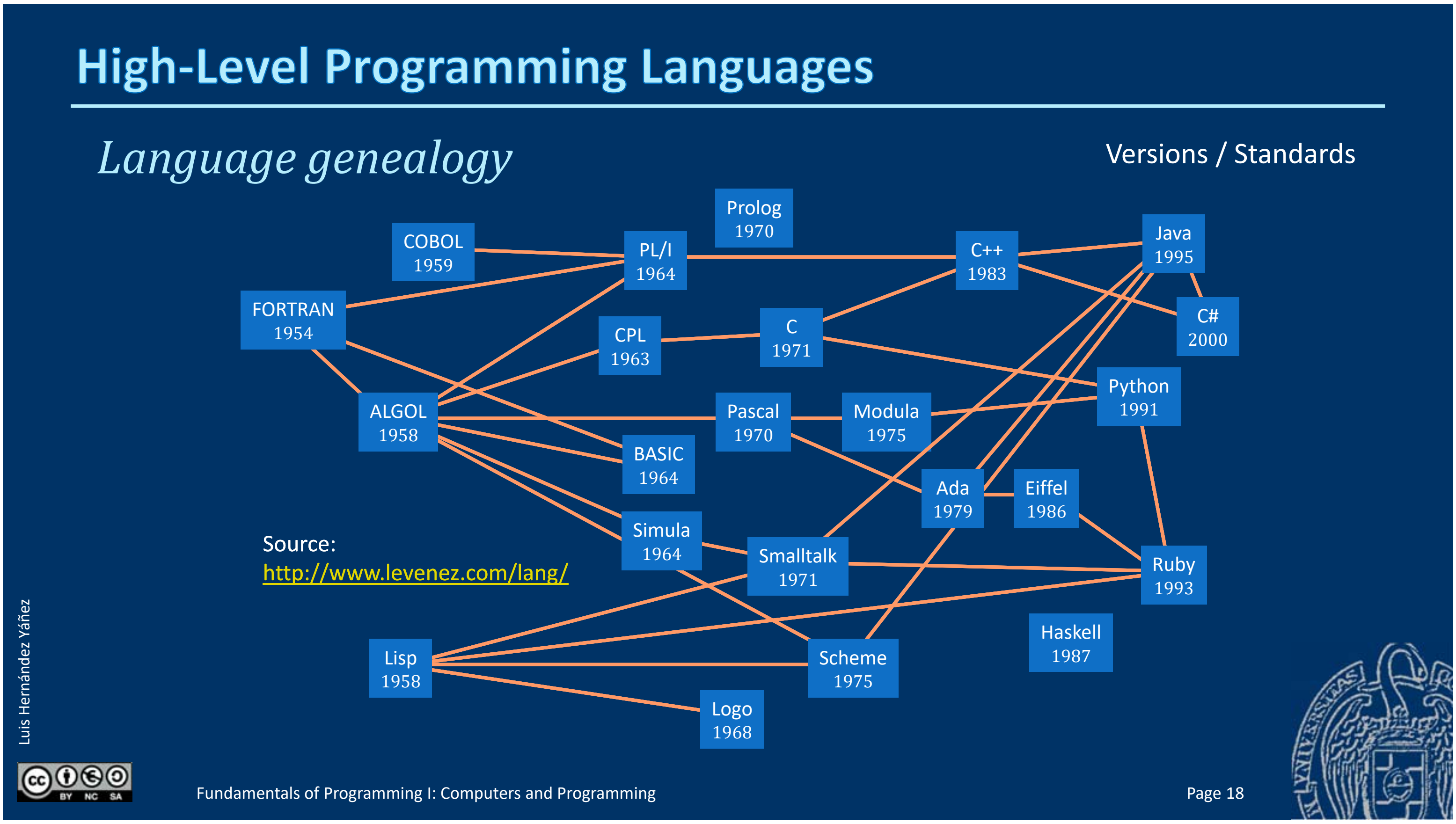
Interpreters:
Translate, link
and execute
one instruction
at a time

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

int main()
{
   cout << "Hola Mundo!" << endl;
   return 0;
}
```

01000101001110100111 00...

For a specific architecture
and operating system

Luis Hernández Yáñez



High-Level Programming Languages

Language genealogy

Versions / Standards

Source:
http://www.levenez.com/lang/

Luis Hernández Yáñez

Fundamentals of Programming I

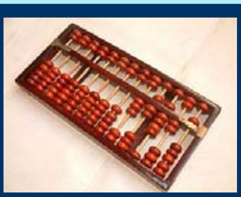
A Little History

Luis Hernández Yáñez

A Little History

Prehistory

Abacus



(Wikipedia)

19th Century

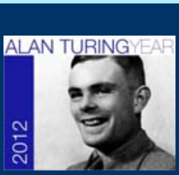
Analytical Machine
(Charles Babbage)

First known programmer!
Lady Ada Lovelace

Luis Hernández Yáñez

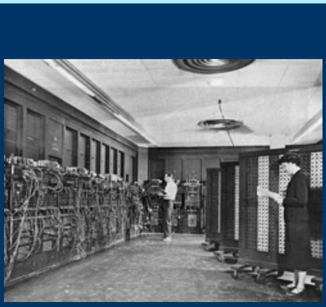
A Little History

20th Century



1936 Turing Machine

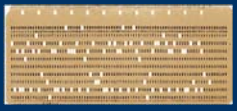
1946 ENIAC: First general-purpose digital computer



ENIAC (Wikipedia)

1947 Transistor

1953 IBM 650: First large-scale industrial computer



1966 ARPANET: Internet predecessor

1967 *Floppy Disk*

1970 UNIX operating system

1972 First computer virus (*Creeper*)
C programming language

1974 TCP protocol (first local network)

Luis Hernández Yáñez

A Little History

1975 Microsoft founded

1976 Apple founded

1979 *Pacman* game

Apple II (Wikipedia)

1981 IBM PC
MS-DOS operating system

1983 C++ programming language

IBM PC (Wikipedia)

1984 CD-ROM

1985 Windows 1.0 operating system

1990 HTML language
World Wide Web

Linux

1991 Linux operating system

Luis Hernández Yáñez

A Little History

1992 Windows 3.1

1995 Java programming language
DVD

1998 Google founded

1999 MSN Messenger

21st Century

2001 Windows XP
Mac OS X

2002 Mozilla Firefox

2007 iPhone

2008 Android . . .

Fundamentals of Programming I

Programming and Software Engineering

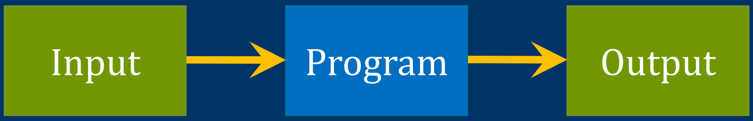
Computer Program

What is Programming?

*To tell to a **very** fast idiot **exactly** what to do*

To specify the structure and behaviour of a program, and test that the program realizes its task properly and with acceptable performance

Program: Transforms input into output



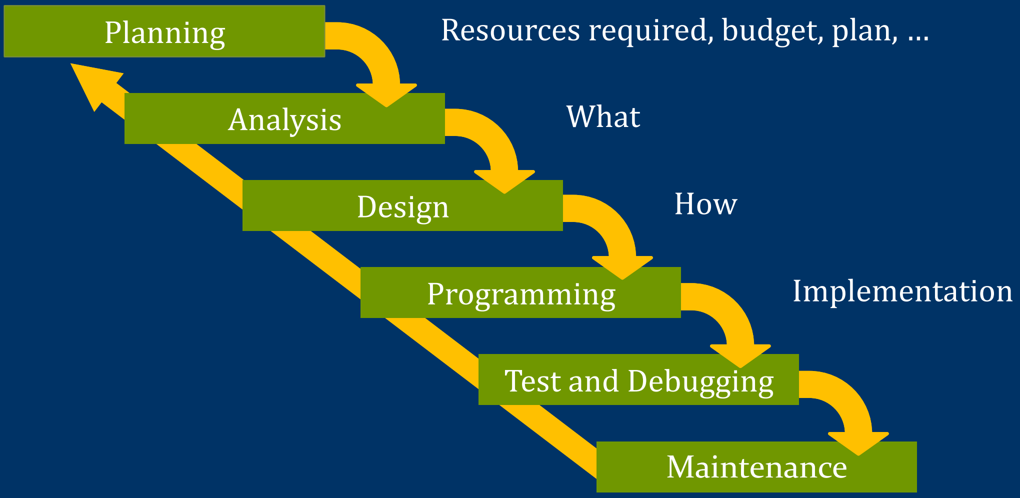
Algorithm: Sequence of steps and operations to be made by the program to solve the problem

The program implements the algorithm in one language

Software Engineering

Programming is just one step in the software development process

"Waterfall" development model:



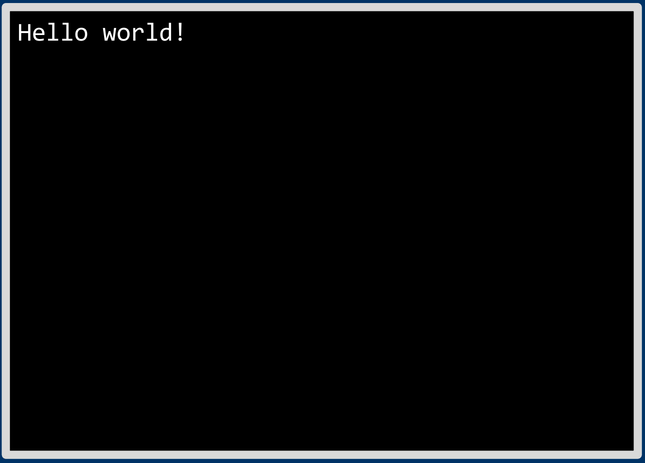
Fundamentals of Programming I

C++ Programming Language

C++ Programming Language

Bjarne Stroustrup (1983)

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

int main()
{
   cout << "Hello world!" << endl;
   // Outputs Hello world!

   return 0;
}
```



```
Hello world!
```

Language Elements

Instructions

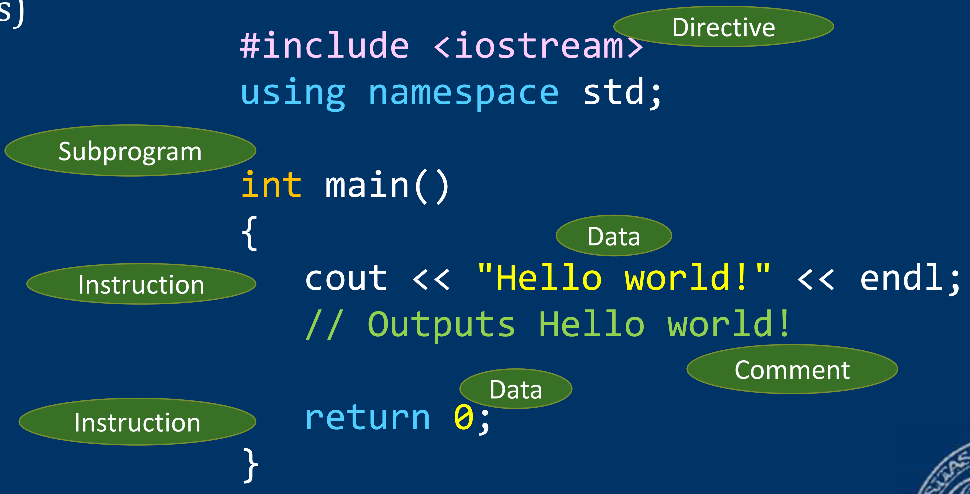
Data: literal data, variables, types

Subprograms (functions)

Comments

Directives

…



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

int main()
{
   cout << "Hello world!" << endl;
   // Outputs Hello world!

   return 0;
}
```

Fundamentals of Programming I

Language Syntax

Language Syntax and Semantics

Syntax: Rules for constructing and sequencing language elements

- ✓ Specification languages

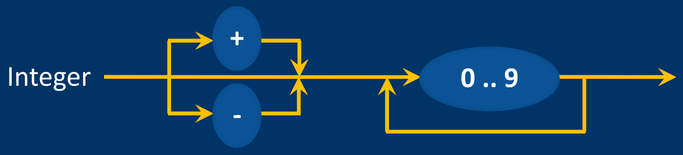
BNF

<integer number> ::= <optional sign><digit sequence>
<optional sign> ::= +|-|<empty>
<digit sequence> ::= <digit>|<digit><digit sequence>
<digit> ::= 0|1|2|3|4|5|6|7|8|9
<empty> ::=

| means OR

+23	✓
-159	✓
1374	✓
1-34	✗
3.4	✗
002	✓

- ✓ Syntax diagrams



Semantics: Meaning of each language element – *What is it for?*

Backus-Naur Form (BNF)

<integer number> ::= <optional sign><digit sequence>
<optional sign> ::= +|-|<empty>
<digit sequence> ::= <digit>|<digit><digit sequence>
<digit> ::= 0|1|2|3|4|5|6|7|8|9
<empty> ::=

+23
<integer number> ::= <optional sign><digit sequence>
::= **+**<digit sequence> ::= **+**<digit><digit sequence>
::= **+2**<digit sequence> ::= **+2**<digit> ::= **+23**

✓

Backus-Naur Form (BNF)

```
<integer number> ::= <optional sign><digit sequence>
<optional sign> ::= +|-|<empty>
<digit sequence> ::= <digit>|<digit><digit sequence>
<digit> ::= 0|1|2|3|4|5|6|7|8|9
<empty> ::=
```

1374

<integer number> ::= <optional sign><digit sequence>

::= <digit sequence> ::= <digit><digit sequence>

::= **1**<digit sequence> ::= **1**<digit><digit sequence> ✓

::= **13**<digit sequence> ::= **13**<digit><digit sequence>

::= **137**<digit sequence> ::= **137**<digit> ::= **1374**

Luis Hernández Yáñez

Backus-Naur Form (BNF)

```
<integer number> ::= <optional sign><digit sequence>
<optional sign> ::= +|-|<empty>
<digit sequence> ::= <digit>|<digit><digit sequence>
<digit> ::= 0|1|2|3|4|5|6|7|8|9
<empty> ::=
```

1-34

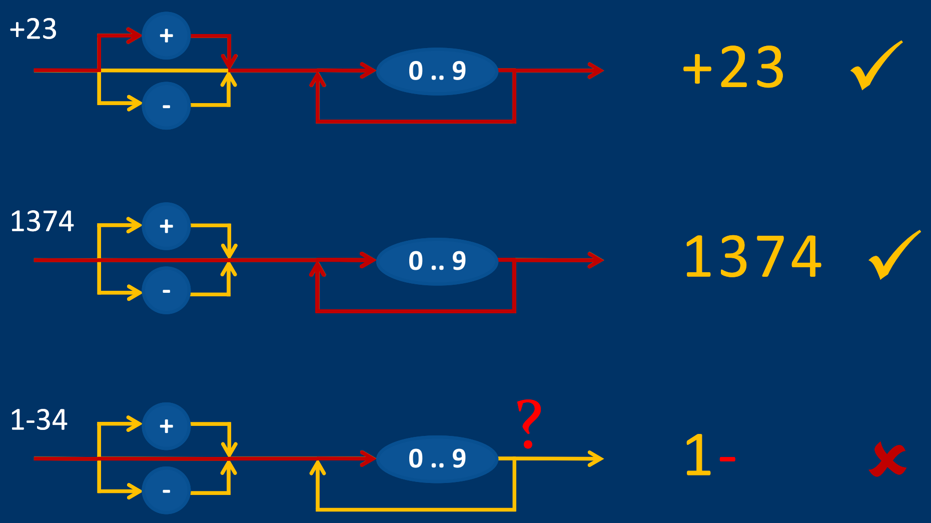
<integer number> ::= <optional sign><digit sequence>

::= <digit sequence> ::= <digit><digit sequence>

::= **1**<digit sequence> ::= **ERROR** (**-** not a <digit>) ✗

Luis Hernández Yáñez

Syntax Diagrams



+23

+

-

0 .. 9

+23 ✓

1374

+

-

0 .. 9

1374 ✓

1-34

+

-

0 .. 9

?

1- ✗

Luis Hernández Yáñez

Fundamentals of Programming I

First C++ Program

Luis Hernández Yáñez

First C++ Program

Hello world!

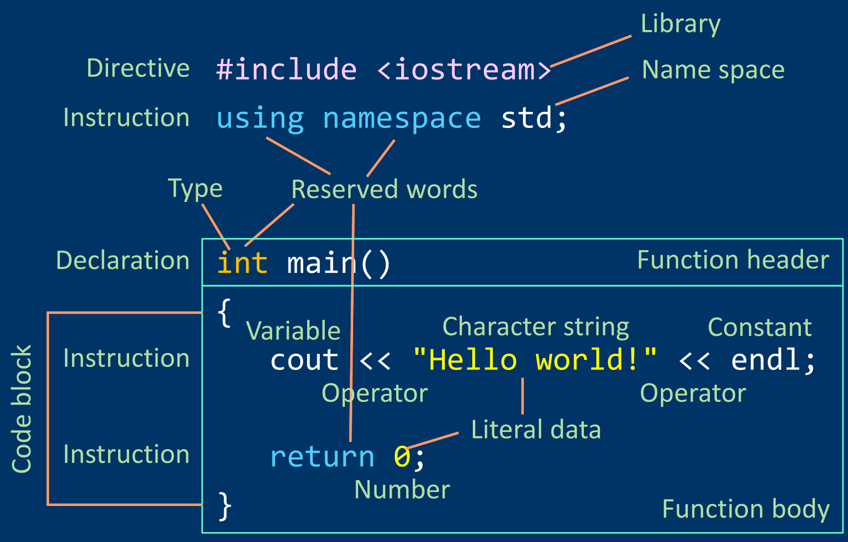
A greeting on the screen:

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

int main()
// main() is where execution starts
{
   cout << "Hello world!" << endl;
   // Outputs Hello world!

   return 0;
}
```

Luis Hernández Yáñez

First C++ Program

Program elements (w/o comments)



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

int main()
{
   cout << "Hello world!" << endl;

   return 0;
}
```

Library

Directive

Name space

Instruction

Type

Reserved words

Declaration

Function header

Variable

Character string

Constant

Code block

Instruction

Operator

Operator

Literal data

Instruction

Number

Function body

Syntactic Coloring

Instructions end with ;

Luis Hernández Yáñez

First C++ Program

Hello world!

Almost everything is infrastructure

Only

```
   cout << "Hello world!" << endl
```

does anything tangible

Infrastructure (notation, libraries and other support)
makes our code simple, complete, reliable and efficient

Style matters!

Luis Hernández Yáñez

Fundamentals of Programming I

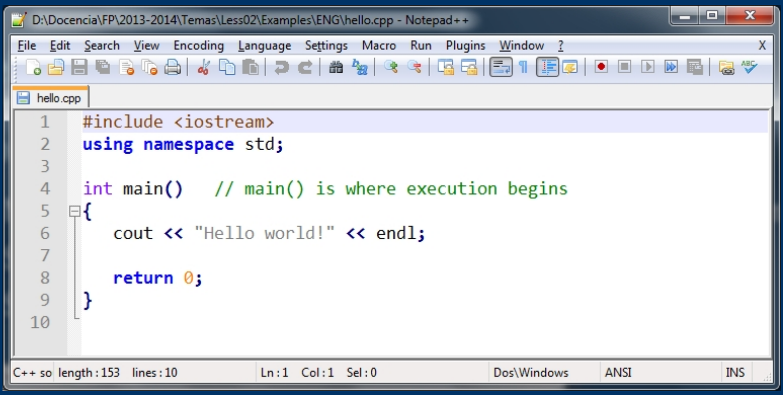
Development Tools

Luis Hernández Yáñez

Development Tools

Editor

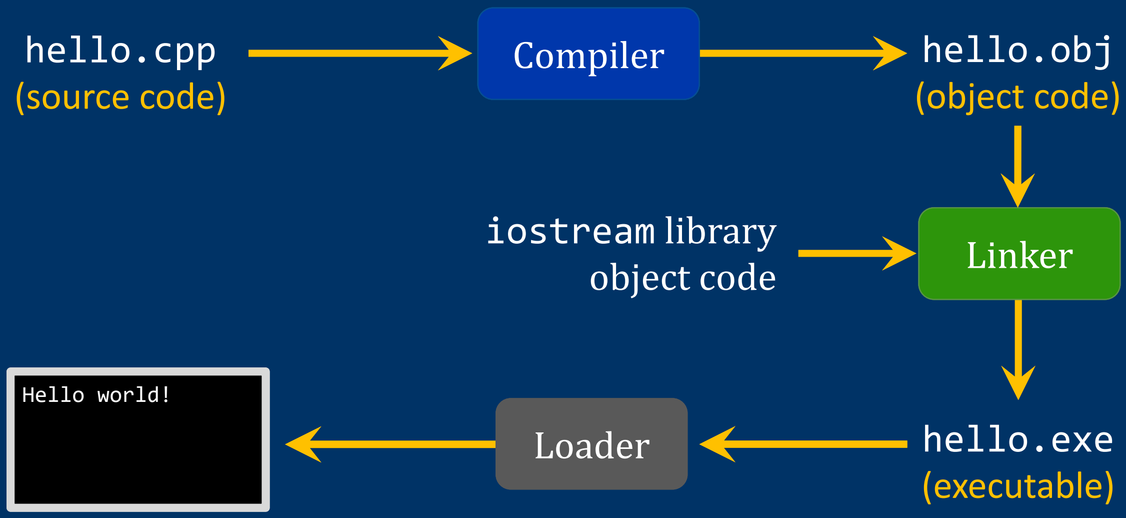
- ✓ Notepad, Wordpad, Gedit, Kwrite, … (simple text, no format)
- ✓ Specific editors: syntactic coloring
- ✓ Suggestion: Notepad++


D:\Docencia\FP\2013-2014\Temas\Less02\Examples\ENG\hello.cpp - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
hello.cpp
C++ so length : 153 lines : 10 Ln : 1 Col : 1 Sel : 0 Dos\Windows ANSI INS

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

int main()   // main() is where execution begins
{
   cout << "Hello world!" << endl;

   return 0;
}

```

Installation and use:
Development Tools
in Virtual Campus

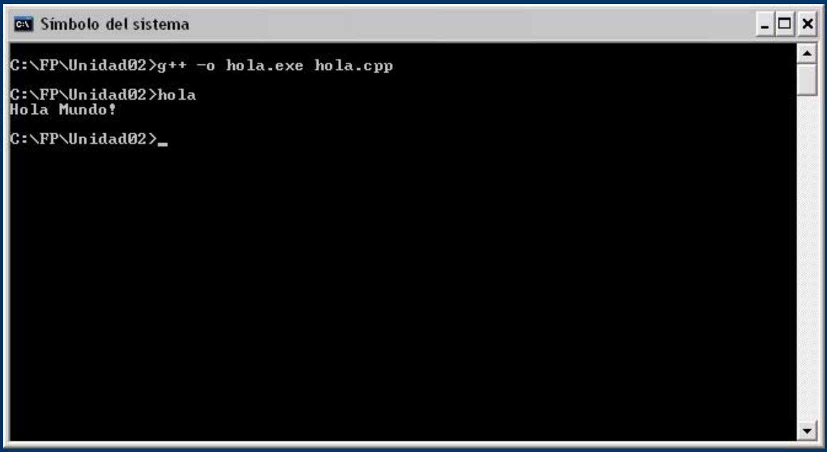
Compiling, Linking, and Executing



Development Tools

Compiler

- ✓ Important: C++ standard
- ✓ Suggestion: GNU G++ (*MinGW* in Windows)



```
C:\FP\Unidad02>g++ -o hola.exe hola.cpp

C:\FP\Unidad02>hola
Hola Mundo!

C:\FP\Unidad02>_
```

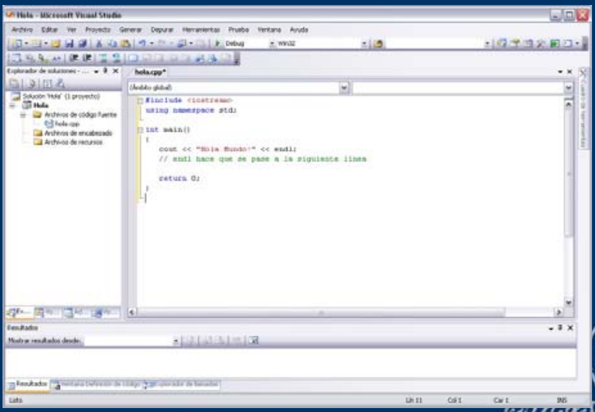
Installation and use:
Development Tools
in Virtual Campus

Luis Hernández Yáñez

Development Tools

Integrated Development Environments

- ✓ Edit, compile and test the program code (and more)
- ✓ Suggestions:
  - — Windows:
    Microsoft Visual Studio (proprietary)
    Eclipse (free)
  - — Linux: Eclipse (free)



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

int main()
{
   cout << "Hola Mundo!" << endl;
   // endl hace que se pase a la siguiente linea

   return 0;
}
```

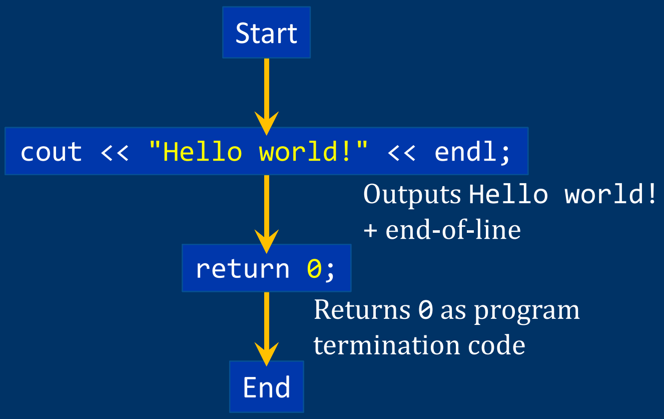
Installation and use:
Development Tools
in Virtual Campus

Luis Hernández Yáñez

The First C++ Program

What does the program do?

- ✓ Program execution always starts in `main()` function
- ✓ Instructions are executed in the order they are in the code



Start

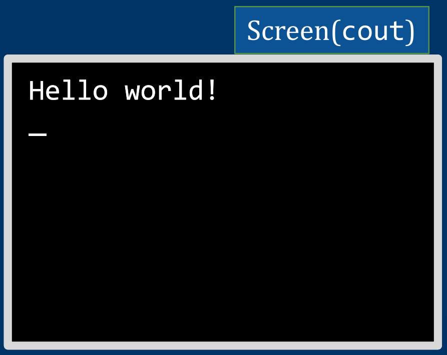
```
cout << "Hello world!" << endl;
```

Outputs `Hello world!`
+ end-of-line

```
return 0;
```

Returns `0` as program
termination code

End



Screen(`cout`)

```
Hello world!
_
```

Fundamentals of Programming I

C++: A Better C

C++: A Better C

The C Language

Dennis M. Ritchie, 1972

- ✓ Middle-level language:
  - — Typical structures of high-level languages
  - — Constructions for machine-level control
- ✓ Simple language (few reserved words)
- ✓ Structured language (but does not allow subprogram nesting)
- ✓ Code and data compartimentalization (blocks and scope)
- ✓ Basic structural component: function (subprogram)
- ✓ Modular programming
- ✓ Case sensitive (lower case and upper case are different)
- ✓ Reserved words (or key-words): lower case

Promote Open Culture!

Creative Commons License

Attribution
You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).

Non commercial
You may not use this work for commercial purposes.

Share alike
If you alter, transform, or build upon this work, you may distribute the resulting work only under the same or similar license to this one.

Click on the upper right image to learn more...