Programinng I

Luis Llana Díaz

Computer Science Departament
Complutense University of Madrid (UCM)

2 de septiembre de 2026

Objetives

Acquire the essential techniques for developing correct, reusable, and efficient small-scale programs, using basic statements, structured control flow, and sub-programs.

- Understand fundamental concepts of programming.
- Learn to design, implement, and debug small-scale programs.
- Master basic control structures (sequence, selection, iteration).
- Develop skills in modular programming using functions and procedures.
- Familiarize with essential data types (numbers, strings, lists, tuples, dictionaries).
- Practice problem-solving using programming.

Methodology

- 4 hours a week (2 days)

Methodology

- 4 hours a week (2 days)
- **Lectures:** (2 hours) Theoretical concepts and small examples.

Methodology

- 4 hours a week (2 days)
- **Lectures:** (2 hours) Theoretical concepts and small examples. Participation is essential for success.

Methodology

- 4 hours a week (2 days)
- **Lectures:** (2 hours) Theoretical concepts and small examples. Participation is essential for success.
- **Exercises:** made by the students.

Methodology

- 4 hours a week (2 days)
- **Lectures:** (2 hours) Theoretical concepts and small examples. Participation is essential for success.
- **Exercises:** made by the students. In class in advance and at home.

Methodology

- 4 hours a week (2 days)
- **Lectures:** (2 hours) Theoretical concepts and small examples. Participation is essential for success.
- **Exercises:** made by the students. In class in advance and at home. 2 hours in laboratory

Methodology

- 4 hours a week (2 days)
- **Lectures:** (2 hours) Theoretical concepts and small examples. Participation is essential for success.
- **Exercises:** made by the students. In class in advance and at home. 2 hours in laboratory
- Evaluation: continuous or final exam

Continuous Evaluation

Continuous Evaluation

Participation:

- Controlled assistance (QR, small exercise at the end of the class)

Continuous Evaluation

Participation:

- Controlled assistance (QR, small exercise at the end of the class)
- 5 % max of missing.

Continuous Evaluation

Participation:

- Controlled assistance (QR, small exercise at the end of the class)
- 5 % max of missing.
- Active participation at class (solving problems at the blackboard).

Continuous Evaluation

Participation:

- Controlled assistance (QR, small exercise at the end of the class)
- 5 % max of missing.
- Active participation at class (solving problems at the blackboard).
- 10 % of the final grade.

Continuous Evaluation

Participation:

- Controlled assistance (QR, small exercise at the end of the class)
- 5 % max of missing.
- Active participation at class (solving problems at the blackboard).
- 10 % of the final grade.

Continuous Evaluation

Participation:
- Controlled assistance (QR, small exercise at the end of the class)
- 5 % max of missing.
- Active participation at class (solving problems at the blackboard).
- 10 % of the final grade.

Practice
- a special exercise (practice).

Continuous Evaluation

Participation:

- Controlled assistance (QR, small exercise at the end of the class)
- 5 % max of missing.
- Active participation at class (solving problems at the blackboard).
- 10 % of the final grade.

Practice

- a special exercise (practice).
- 3 people groups.

Continuous Evaluation

Participation:

- Controlled assistance (QR, small exercise at the end of the class)
- 5 % max of missing.
- Active participation at class (solving problems at the blackboard).
- 10 % of the final grade.

Practice

- a special exercise (practice).
- 3 people groups.
- Oral presentation.

Continuous Evaluation

Participation:

- Controlled assistance (QR, small exercise at the end of the class)
- 5 % max of missing.
- Active participation at class (solving problems at the blackboard).
- 10 % of the final grade.

Practice

- a special exercise (practice).
- 3 people groups.
- Oral presentation.
- 20 % of the final grade.

Continuous Evaluation

Participation:

- Controlled assistance (QR, small exercise at the end of the class)
- 5 % max of missing.
- Active participation at class (solving problems at the blackboard).
- 10 % of the final grade.

Practice

- a special exercise (practice).
- 3 people groups.
- Oral presentation.
- 20 % of the final grade.

Continuous Evaluation

Participation:
- Controlled assistance (QR, small exercise at the end of the class)
- 5 % max of missing.
- Active participation at class (solving problems at the blackboard).
- 10 % of the final grade.

Practice
- a special exercise (practice).
- 3 people groups.
- Oral presentation.
- 20 % of the final grade.

Partial Exams
- In paper.

Continuous Evaluation

Participation:
- Controlled assistance (QR, small exercise at the end of the class)
- 5 % max of missing.
- Active participation at class (solving problems at the blackboard).
- 10 % of the final grade.

Practice
- a special exercise (practice).
- 3 people groups.
- Oral presentation.
- 20 % of the final grade.

Partial Exams
- In paper.
- 2 partial exams (40 %, 60 %).

Continuous Evaluation

Participation:
- Controlled assistance (QR, small exercise at the end of the class)
- 5 % max of missing.
- Active participation at class (solving problems at the blackboard).
- 10 % of the final grade.

Practice
- a special exercise (practice).
- 3 people groups.
- Oral presentation.
- 20 % of the final grade.

Partial Exams
- In paper.
- 2 partial exams (40 %, 60 %).
- Minimal grade each 4 out of 10.

Continuous Evaluation

Participation:
- Controlled assistance (QR, small exercise at the end of the class)
- 5 % max of missing.
- Active participation at class (solving problems at the blackboard).
- 10 % of the final grade.

Practice
- a special exercise (practice).
- 3 people groups.
- Oral presentation.
- 20 % of the final grade.

Partial Exams
- In paper.
- 2 partial exams (40 %, 60 %).
- Minimal grade each 4 out of 10.
- 70 % of the final grade.

Way to success

My responsability. To teach you the concepts.

Way to success

My responsability. To teach you the concepts. I have prepared material.

Way to success

My responsability. To teach you the concepts. I have prepared material.

My wish. You to learn.

Way to success

My responsability. To teach you the concepts. I have prepared material.

My wish. You to learn. I want to solve all your doubts.

Way to success

My responsability. To teach you the concepts. I have prepared material.

My wish. You to learn. I want to solve all your doubts.

Your responsability. To learn.

Way to success

My responsability. To teach you the concepts. I have prepared material.

My wish. You to learn. I want to solve all your doubts.

Your responsability. To learn.

Your wish. To pass the subject with good grades.

Way to success

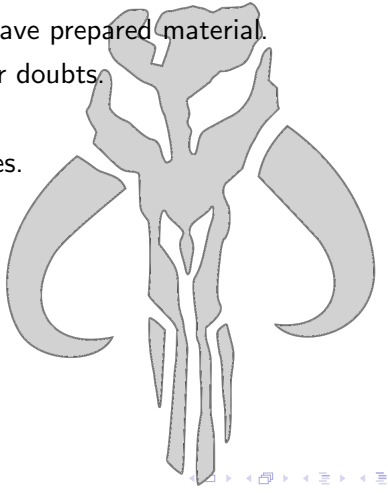
My responsability. To teach you the concepts. I have prepared material.

My wish. You to learn. I want to solve all your doubts.

Your responsability. To learn.

Your wish. To pass the subject with good grades.

This is the way



Way to success

My responsability. To teach you the concepts. I have prepared material.

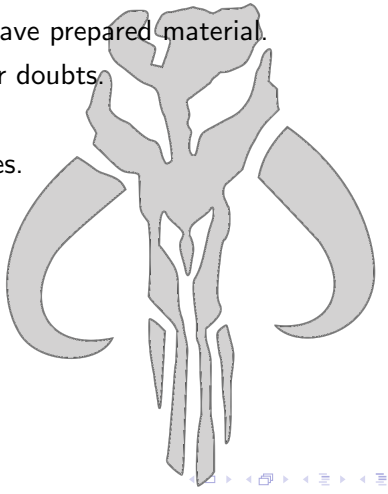
My wish. You to learn. I want to solve all your doubts.

Your responsability. To learn.

Your wish. To pass the subject with good grades.

This is the way

- Work hard (you and me).



Way to success

My responsability. To teach you the concepts. I have prepared material.

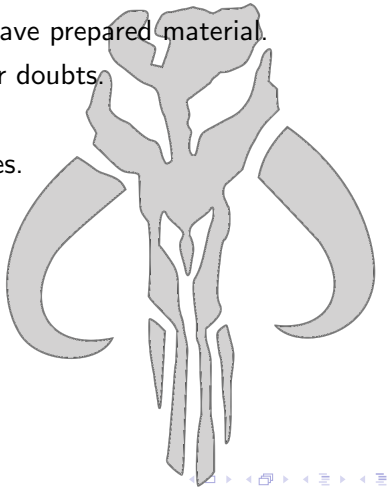
My wish. You to learn. I want to solve all your doubts.

Your responsability. To learn.

Your wish. To pass the subject with good grades.

This is the way

- Work hard (you and me).
- Make the exercises.



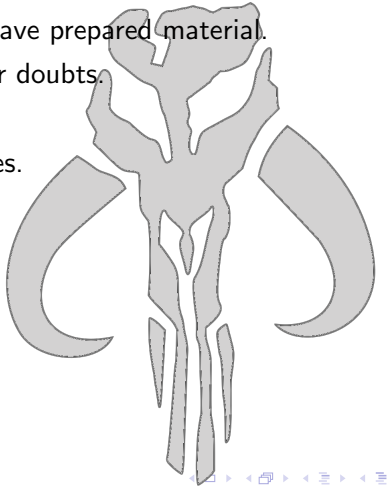
Way to success

My responsability. To teach you the concepts. I have prepared material.

My wish. You to learn. I want to solve all your doubts.

Your responsability. To learn.

Your wish. To pass the subject with good grades.



This is the way

- Work hard (you and me).
- Make the exercises.
- Ask all the doubts (you).

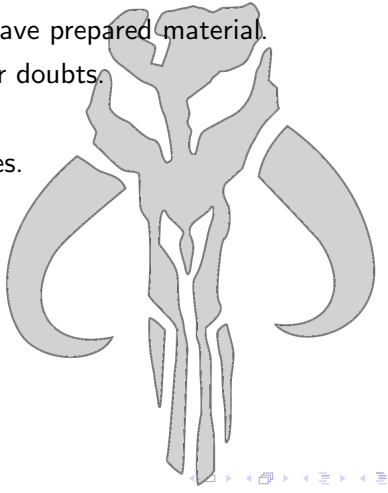
Way to success

My responsability. To teach you the concepts. I have prepared material.

My wish. You to learn. I want to solve all your doubts.

Your responsability. To learn.

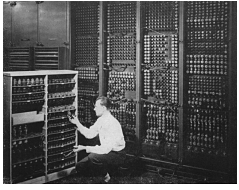
Your wish. To pass the subject with good grades.



This is the way

- Work hard (you and me).
- Make the exercises.
- Ask all the doubts (you).
- Solve your questions (me).

What is a Computer?



Source: https://en.wikipedia.org/wiki/Computer

What is a Computer?

Definition

A **computer** is a machine that is

Source: https://en.wikipedia.org/wiki/Computer

What is a Computer?

Definition

A **computer** is a machine that is

- programmable and executes a series of commands,

Source: https://en.wikipedia.org/wiki/Computer

What is a Computer?

Definition

A **computer** is a machine that is

- programmable and executes a series of commands,
- processes input data,

Source: https://en.wikipedia.org/wiki/Computer

What is a Computer?

Definition

A **computer** is a machine that is

- programmable and executes a series of commands,
- processes input data,
- obtaining useful information,

Source: https://en.wikipedia.org/wiki/Computer

What is a Computer?

Definition

A **computer** is a machine that is

- programmable and executes a series of commands,
- processes input data,
- obtaining useful information,
- which is subsequently sent to output units.

Source: https://en.wikipedia.org/wiki/Computer

What is a Computer?

Definition

A **computer** is a machine that is

- programmable and executes a series of commands,
- processes input data,
- obtaining useful information,
- which is subsequently sent to output units.

Components

Source: https://en.wikipedia.org/wiki/Computer

What is a Computer?

Definition

A **computer** is a machine that is

- programmable and executes a series of commands,
- processes input data,
- obtaining useful information,
- which is subsequently sent to output units.

Components

Hardware Physical structure (electronic circuits, cables, cabinet, keyboard, mouse, etc.).

Source: https://en.wikipedia.org/wiki/Computer

What is a Computer?

Definition

A **computer** is a machine that is

- programmable and executes a series of commands,
- processes input data,
- obtaining useful information,
- which is subsequently sent to output units.

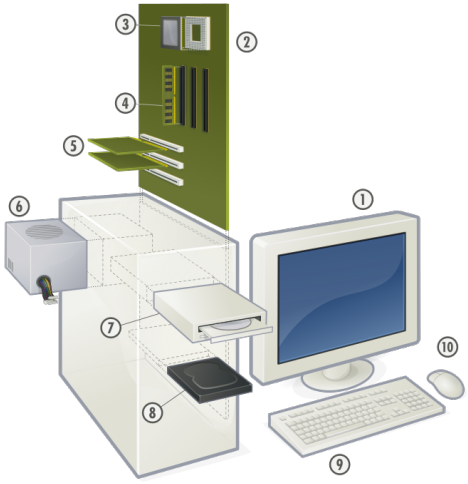
Components

Hardware Physical structure (electronic circuits, cables, cabinet, keyboard, mouse, etc.).

Software Intangible part (programs, data, information, documentation, etc.).

Source: https://en.wikipedia.org/wiki/Computer

Physical Structure



1. Monitor.
2. Motherboard (Main board).
3. Microprocessor (CPU).
4. RAM Memory.
5. Expansion cards and slots.
6. Power supply.
7. Optical disc drive (CD; DVD; BD).
8. Hard disk drive or Solid-state drive.
9. Keyboard.
10. Mouse.

Computer Capacity and Performance

Number of Cores Indicates how many CPUs it has (parallel processing). A modern personal computer can have 6 cores.

Computer Capacity and Performance

Number of Cores Indicates how many CPUs it has (parallel processing). A modern personal computer can have 6 cores.

CPU Speed The number of basic instructions per second.

Computer Capacity and Performance

Number of Cores Indicates how many CPUs it has (parallel processing). A modern personal computer can have 6 cores.

CPU Speed The number of basic instructions per second.

Memory Capacity

Computer Capacity and Performance

Number of Cores Indicates how many CPUs it has (parallel processing). A modern personal computer can have 6 cores.

CPU Speed The number of basic instructions per second.

Memory Capacity

- Internal Memory (RAM)

Computer Capacity and Performance

Number of Cores Indicates how many CPUs it has (parallel processing). A modern personal computer can have 6 cores.

CPU Speed The number of basic instructions per second.

Memory Capacity

- Internal Memory (RAM)
- Storage Space (Hard Drive, SSD)

CPU Speed

- Basic instructions, e.g., adding two numbers.

CPU Speed

- Basic instructions, e.g., adding two numbers.
- How many basic instructions per second it is capable of executing.

CPU Speed

- Basic instructions, e.g., adding two numbers.
- How many basic instructions per second it is capable of executing.
- Speed: 1 Hz = 1 instruction per second.

CPU Speed

- Basic instructions, e.g., adding two numbers.
- How many basic instructions per second it is capable of executing.
- Speed: 1 Hz = 1 instruction per second.
- Apollo Guidance Computer: 1 core @ 2.048 MHz.

CPU Speed

- Basic instructions, e.g., adding two numbers.
- How many basic instructions per second it is capable of executing.
- Speed: 1 Hz = 1 instruction per second.
- Apollo Guidance Computer: 1 core @ 2.048 MHz.
- Commodore 64 (1980s): 1 core @ 1 MHz.

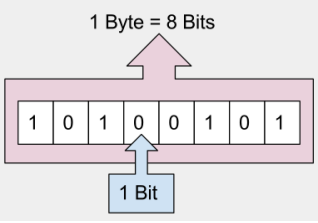
CPU Speed

- Basic instructions, e.g., adding two numbers.
- How many basic instructions per second it is capable of executing.
- Speed: 1 Hz = 1 instruction per second.
- Apollo Guidance Computer: 1 core @ 2.048 MHz.
- Commodore 64 (1980s): 1 core @ 1 MHz.
- Intel Core i7: 6 cores @ 3.3 GHz. ($1G = 10^9$).

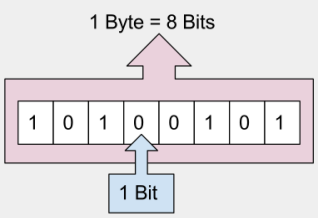
CPU Speed

- Basic instructions, e.g., adding two numbers.
- How many basic instructions per second it is capable of executing.
- Speed: 1 Hz = 1 instruction per second.
- Apollo Guidance Computer: 1 core @ 2.048 MHz.
- Commodore 64 (1980s): 1 core @ 1 MHz.
- Intel Core i7: 6 cores @ 3.3 GHz. ($1G = 10^9$).
- El Capitan (Current top supercomputer): 1,051,392 cores @ 1.8 GHz.

Storage: Memory Magnitudes

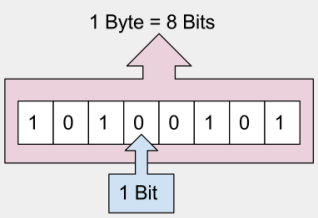


Storage: Memory Magnitudes



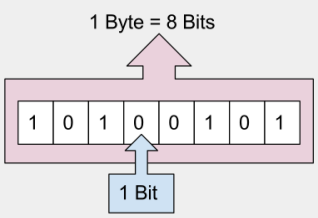
Kilobyte (KB) $2^{10} \approx 10^3$.

Storage: Memory Magnitudes



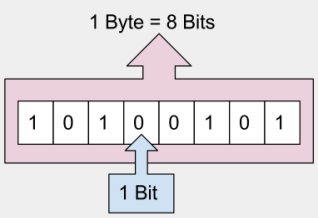
Kilobyte (KB) $2^{10} \approx 10^3$.
Megabyte (MB) $2^{20} \approx 10^6$.

Storage: Memory Magnitudes



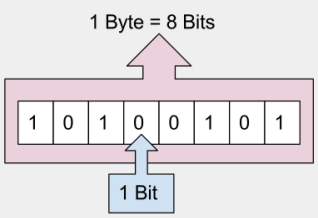
Kilobyte (KB) $2^{10} \approx 10^3$.
Megabyte (MB) $2^{20} \approx 10^6$.
Gigabyte (GB) $2^{30} \approx 10^9$.

Storage: Memory Magnitudes



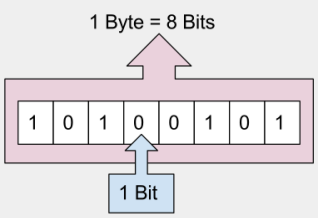
Kilobyte (KB) $2^{10} \approx 10^3$.
Megabyte (MB) $2^{20} \approx 10^6$.
Gigabyte (GB) $2^{30} \approx 10^9$.
Terabyte (TB) $2^{40} \approx 10^{12}$.

Storage: Memory Magnitudes



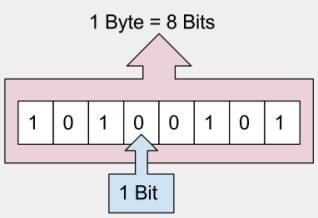
Kilobyte (KB) $2^{10} \approx 10^3$.
Megabyte (MB) $2^{20} \approx 10^6$.
Gigabyte (GB) $2^{30} \approx 10^9$.
Terabyte (TB) $2^{40} \approx 10^{12}$.
Petabyte (PB) $2^{50} \approx 10^{15}$.

Storage: Memory Magnitudes



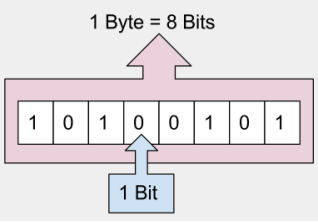
Kilobyte (KB) $2^{10} \approx 10^3$.
Megabyte (MB) $2^{20} \approx 10^6$.
Gigabyte (GB) $2^{30} \approx 10^9$.
Terabyte (TB) $2^{40} \approx 10^{12}$.
Petabyte (PB) $2^{50} \approx 10^{15}$.
Exabyte (EB) $2^{60} \approx 10^{18}$.

Storage: Memory Magnitudes



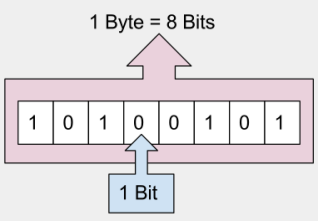
Kilobyte (KB) $2^{10} \approx 10^3$. Don Quixote (book): 850 KB
Megabyte (MB) $2^{20} \approx 10^6$.
Gigabyte (GB) $2^{30} \approx 10^9$.
Terabyte (TB) $2^{40} \approx 10^{12}$.
Petabyte (PB) $2^{50} \approx 10^{15}$.
Exabyte (EB) $2^{60} \approx 10^{18}$.

Storage: Memory Magnitudes



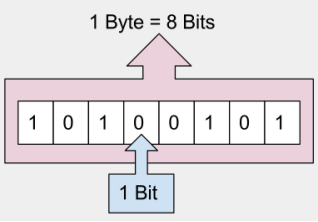
Kilobyte (KB) $2^{10} \approx 10^3$. Don Quixote (book): 850 KB
Megabyte (MB) $2^{20} \approx 10^6$. Photo: 3 MB (8Mpx). 1 CD: 800 MB
Gigabyte (GB) $2^{30} \approx 10^9$.
Terabyte (TB) $2^{40} \approx 10^{12}$.
Petabyte (PB) $2^{50} \approx 10^{15}$.
Exabyte (EB) $2^{60} \approx 10^{18}$.

Storage: Memory Magnitudes



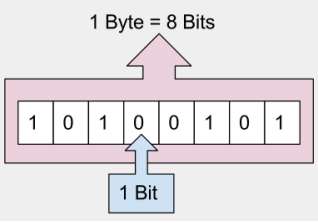
Kilobyte (KB) $2^{10} \approx 10^3$. Don Quixote (book): 850 KB
Megabyte (MB) $2^{20} \approx 10^6$. Photo: 3 MB (8Mpx). 1 CD: 800 MB
Gigabyte (GB) $2^{30} \approx 10^9$. DVD: 4.7 GB. Wikipedia: 120 GB
Terabyte (TB) $2^{40} \approx 10^{12}$.
Petabyte (PB) $2^{50} \approx 10^{15}$.
Exabyte (EB) $2^{60} \approx 10^{18}$.

Storage: Memory Magnitudes



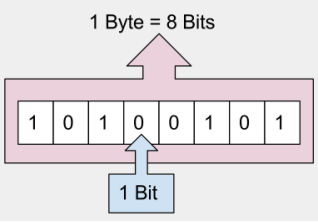
Kilobyte (KB) $2^{10} \approx 10^3$. Don Quixote (book): 850 KB
Megabyte (MB) $2^{20} \approx 10^6$. Photo: 3 MB (8Mpx). 1 CD: 800 MB
Gigabyte (GB) $2^{30} \approx 10^9$. DVD: 4.7 GB. Wikipedia: 120 GB
Terabyte (TB) $2^{40} \approx 10^{12}$. +200 DVDs. Facebook: 750 TB/day
Petabyte (PB) $2^{50} \approx 10^{15}$.
Exabyte (EB) $2^{60} \approx 10^{18}$.

Storage: Memory Magnitudes



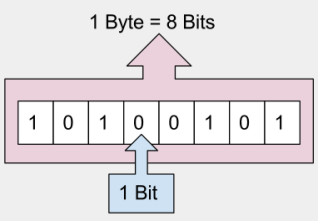
Kilobyte (KB) $2^{10} \approx 10^3$. Don Quixote (book): 850 KB
Megabyte (MB) $2^{20} \approx 10^6$. Photo: 3 MB (8Mpx). 1 CD: 800 MB
Gigabyte (GB) $2^{30} \approx 10^9$. DVD: 4.7 GB. Wikipedia: 120 GB
Terabyte (TB) $2^{40} \approx 10^{12}$. +200 DVDs. Facebook: 750 TB/day
Petabyte (PB) $2^{50} \approx 10^{15}$. Netflix: 3.14 PB. Google Maps: 20 PB
Exabyte (EB) $2^{60} \approx 10^{18}$.

Storage: Memory Magnitudes



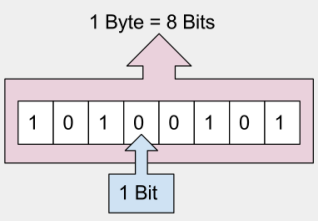
Kilobyte (KB) $2^{10} \approx 10^3$. Don Quixote (book): 850 KB
Megabyte (MB) $2^{20} \approx 10^6$. Photo: 3 MB (8Mpx). 1 CD: 800 MB
Gigabyte (GB) $2^{30} \approx 10^9$. DVD: 4.7 GB. Wikipedia: 120 GB
Terabyte (TB) $2^{40} \approx 10^{12}$. +200 DVDs. Facebook: 750 TB/day
Petabyte (PB) $2^{50} \approx 10^{15}$. Netflix: 3.14 PB. Google Maps: 20 PB
Exabyte (EB) $2^{60} \approx 10^{18}$. Google: 10 EB on disk

Storage: Memory Magnitudes



Kilobyte (KB) $2^{10} \approx 10^3$. Don Quixote (book): 850 KB
Megabyte (MB) $2^{20} \approx 10^6$. Photo: 3 MB (8Mpx). 1 CD: 800 MB
Gigabyte (GB) $2^{30} \approx 10^9$. DVD: 4.7 GB. Wikipedia: 120 GB
Terabyte (TB) $2^{40} \approx 10^{12}$. +200 DVDs. Facebook: 750 TB/day
Petabyte (PB) $2^{50} \approx 10^{15}$. Netflix: 3.14 PB. Google Maps: 20 PB
Exabyte (EB) $2^{60} \approx 10^{18}$. Google: 10 EB on disk
Zettabytes (ZB), Yottabytes (YB)...

Storage: Memory Magnitudes



Kilobyte (KB) $2^{10} \approx 10^3$. Don Quixote (book): 850 KB
Megabyte (MB) $2^{20} \approx 10^6$. Photo: 3 MB (8Mpx). 1 CD: 800 MB
Gigabyte (GB) $2^{30} \approx 10^9$. DVD: 4.7 GB. Wikipedia: 120 GB
Terabyte (TB) $2^{40} \approx 10^{12}$. +200 DVDs. Facebook: 750 TB/day
Petabyte (PB) $2^{50} \approx 10^{15}$. Netflix: 3.14 PB. Google Maps: 20 PB
Exabyte (EB) $2^{60} \approx 10^{18}$. Google: 10 EB on disk
Zettabytes (ZB), Yottabytes (YB)...

`http://www.bluebulbprojects.com/MeasureOfThings/`

Memory Evolution

- Apollo Guidance Computer: 2 KB (RAM), 36 KB (ROM).

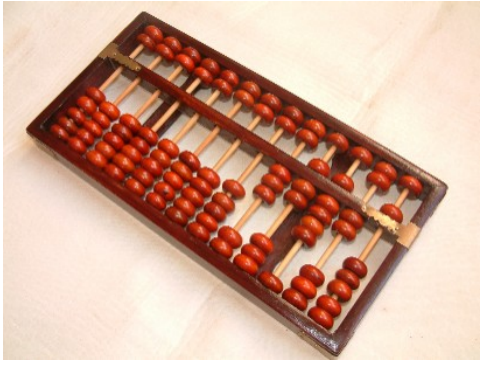
Memory Evolution

- Apollo Guidance Computer: 2 KB (RAM), 36 KB (ROM).
- Commodore 64: 64 KB (RAM), 16 KB (ROM).

Memory Evolution

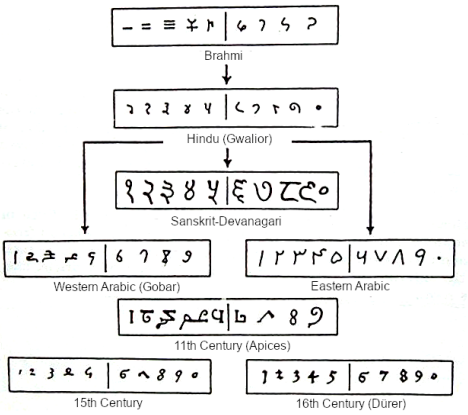
- Apollo Guidance Computer: 2 KB (RAM), 36 KB (ROM).
- Commodore 64: 64 KB (RAM), 16 KB (ROM).
- Modern PC: 32 GB RAM, 1 TB (Hard Drive/SSD).

Historical Perspective I



- More than 5,000 years ago.
- Addition, subtraction, multiplication, division, and square roots.

Historical Perspective II



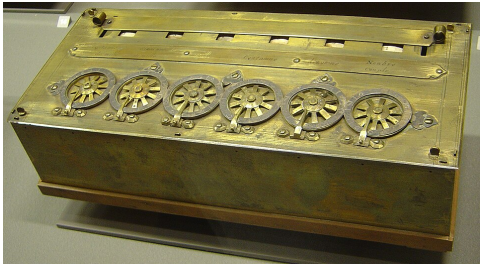
- Indo-Arabic numerals.
- Facilitated mathematical operations.
- Faster and more reliable.

Number one story: `https://youtu.be/jrLQW1vQklE?si=Rv_RwWJzURBCMNpV`

Historical Perspective III

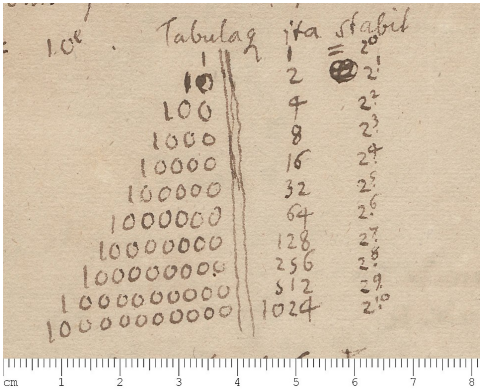
- Muḥammad Al-Khwarizmi, Latinized as Algorithmi.
- 9th-century Persian mathematician.
- Father of Algebra.
- Precursor to modern algorithms.

Historical Perspective IV



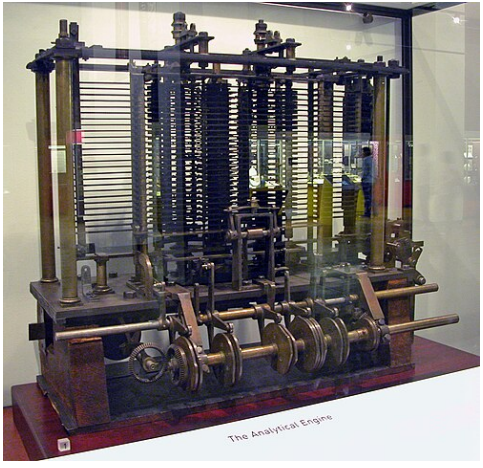
- The Pascaline.
- Blaise Pascal (at age 19), 17th Century.
- Mechanical addition using gear wheels.

Historical Perspective V



- Binary System.
- Leibniz, 17th Century.
- Influenced by ancient systems in India and China (I-Ching).

Historical Perspective VI

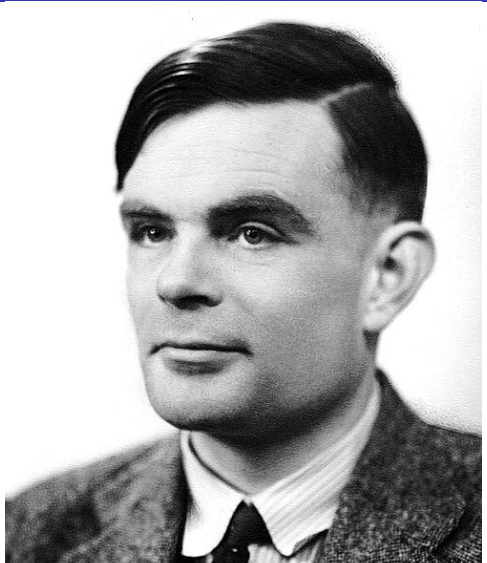


- Charles Babbage's Analytical Engine, 19th Century.
- Never fully built due to the technical complexity of the era.

Historical Perspective VII

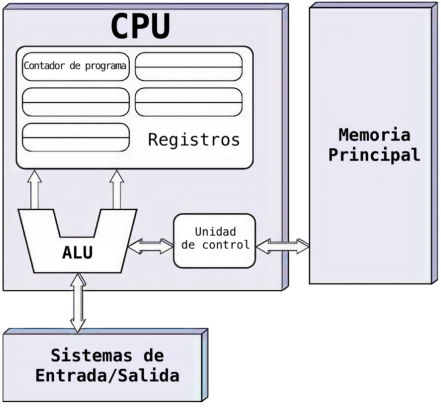
- Ada Lovelace (Augusta Ada Byron).
- The world's first computer programmer.
- Wrote programs for Babbage's machine.

Historical Perspective VIII



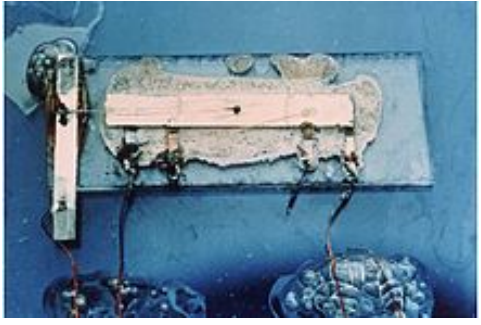
- Mathematician, Father of Modern Computing.
- Cracked the Enigma (Nazi's encrypting machine, II World World) code with the Bombe machine.
- Convicted for his sexual orientation; pardoned decades after his death.

Historical Perspective IX



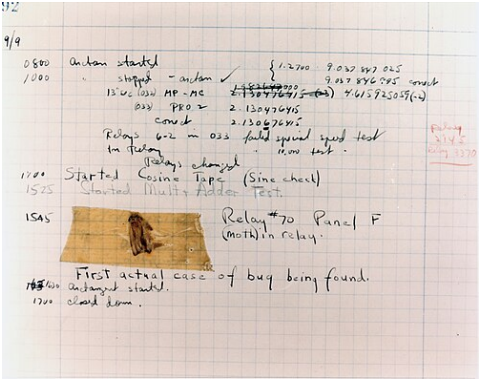
- John von Neumann (1945).
- Defined the architecture used in current computers.

Historical Perspective X



- The Integrated Circuit (IC).
- Jack S. Kilby, 1958.

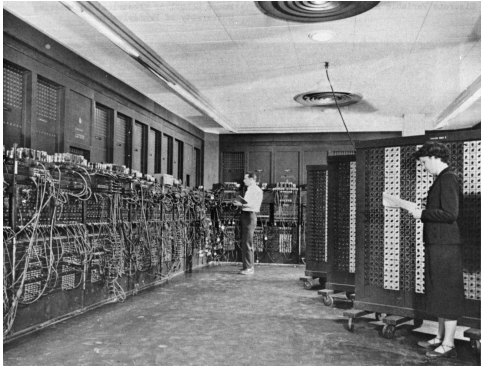
Historical Perspective XI



- The first documented "Bug" was a literal moth.
- 1947: A hardware failure caused by an insect.

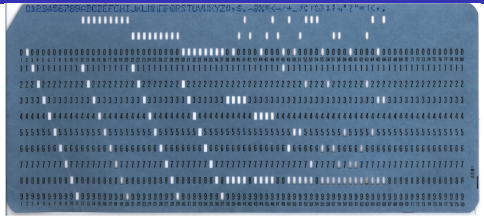
Historical Perspective XII

ENIAC



- 1945: First general-purpose programmable electronic computer.
- Turing Complete: Capable of solving any computable problem.
- Cost $487,000 (approx. $6.9 million today).
- Size: 2m x 1m x 30m. Weight: 20 Tons.
- Speed: 5 kHz.
- Storage: Punched cards. Output: Printer.

Historical Perspective XIII



- Each card represents a line of 80 characters.
- A program is a stack of these punched cards.

Historical Perspective XIV

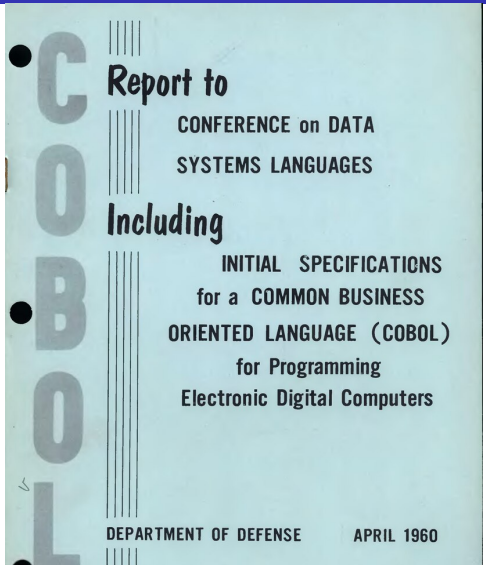
IBM 360 (1964)

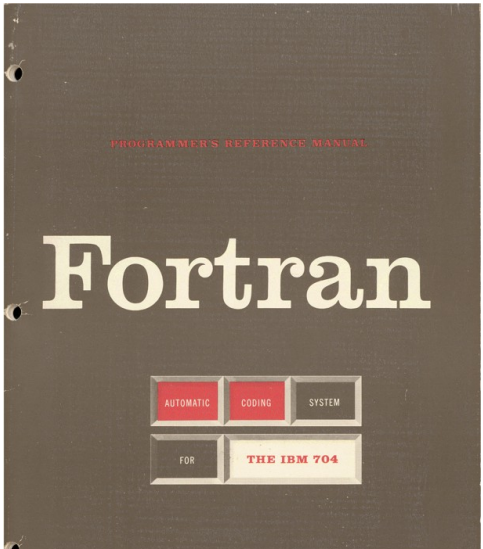
- First large-scale commercial computers.
- Speed: 34.5 kHz. Memory: 8–64 KB.

Historical Perspective XV

Languages of the 60s: COBOL, FORTRAN

Historical Perspective XVI





What is Programming?

Programming is the process of designing, writing, testing, debugging, and maintaining the source code of computer programs.

- **Algorithm:** A finite, ordered sequence of well-defined instructions that solves a problem.
- **Source Code:** The set of instructions written in a programming language.
- **Programming Language:** A formal language designed to communicate instructions to a machine.

Why Python?

- **Easy to learn and read:** Clear and concise syntax.
- **Versatile:** Wide range of applications (Web, AI, Data Science, Automation, etc.).
- **Large community:** Extensive support and resources.
- **Multi-paradigm:** Supports Object-Oriented, Functional, and Structured programming.
- **Extensive libraries:** A rich ecosystem of modules.

Your First Program: "Hello World"

```
print("Hello World")
```

- The `print()` function is used to display text on the console.
- Text strings are enclosed in either double (") or single (') quotes.

Comments

```
# This is a single-line comment
print("Hello") # This is a comment at the end of a line

"""
This␣is␣a␣multi-line
comment␣(docstring).
It␣is␣used␣to␣explain␣blocks␣of␣code.
"""
print("World")
```

- Comments are ignored by the Python interpreter.
- They help make the code more readable and understandable.

Variables

```
name = "Alice"
age = 30
height = 1.75
is_student = True

print(name)
print(age)
print(height)
print(is_student)
```

- Variables are containers for storing data values.
- Python is dynamically typed: you do not need to declare the variable type.

Basic Data Types

- **Integers (`int`):** Whole numbers, positive or negative (e.g., `10`, `-5`).
- **Floats (`float`):** Numbers with a decimal point (e.g., `3.14`, `-0.5`).
- **Strings (`str`):** Sequences of characters (e.g., `"Hello"`, `'Python'`).
- **Booleans (`bool`):** Truth values (`True`, `False`).

Comparison Operators

```
x = 10
y = 12

print(f"x == y: {x == y}")   # False
print(f"x != y: {x != y}")   # True
print(f"x > y: {x > y}")     # False
print(f"x < y: {x < y}")     # True
print(f"x >= y: {x >= y}")   # False
print(f"x <= y: {x <= y}")   # True
```

Logical Operators

```
p = True
q = False

print(f"p and q: {p and q}") # False
print(f"p or q: {p or q}")   # True
print(f"not p: {not p}")     # False
```

Entrada de Datos

```
nombre = input("Ingresa␣tu␣nombre:␣")
edad_str = input("Ingresa␣tu␣edad:␣")
edad = int(edad_str) # Convertir a entero

print(f"Hola,␣{nombre}.␣Tienes␣{edad}␣ñaos.")
```

- La función `input()` lee una línea de texto del usuario.
- Siempre devuelve una cadena de texto (`str`).
- Es necesario convertir a otros tipos de datos si se requiere (ej. `int()`, `float()`).

User Input

```
name = input("Enter your name: ")
age_str = input("Enter your age: ")
age = int(age_str) # Convert to integer

print(f"Hello, {name}. You are {age} years old.")
```

- The `input()` function reads a line of text from the user.
- It always returns a string (`str`).
- You must convert it to other data types if needed (e.g., `int()`, `float()`).

Control Structures: `if-elif-else`

```
temperature = 25

if temperature > 30:
    print("It is very hot.")
elif temperature > 20:
    print("The temperature is pleasant.")
else:
    print("It is cold.")
```

- Allows executing different blocks of code based on conditions.
- Indentation is crucial in Python to define code blocks.

Estructuras de Control: `while` Loop

```
contador = 0
while contador < 5:
  print(f"Contador: {contador}")
  contador += 1 # contador = contador + 1
```

- Repite un bloque de código mientras una condición sea verdadera.
- Es importante asegurarse de que la condición eventualmente se vuelva falsa para evitar bucles infinitos.

Control Structures: `while` Loop

```
counter = 0
while counter < 5:
    print(f"Counter: {counter}")
    counter += 1 # equivalent to counter = counter + 1
```

- Repeats a block of code as long as a condition is true.
- It is important to ensure that the condition eventually becomes false to avoid infinite loops.

Estructuras de Control: `for` Loop

```
frutas = ["manzana", "banana", "cereza"]
for fruta in frutas:
  print(fruta)

for i in range(5): # i toma valores 0, 1, 2, 3, 4
  print(f"úNmero: {i}")
```

- Itera sobre elementos de una secuencia (listas, cadenas, rangos, etc.).
- La función `range(n)` genera una secuencia de números desde 0 hasta n-1.

Control Structures: `for` Loop

```
fruits = ["apple", "banana", "cherry"]
for fruit in fruits:
    print(fruit)

for i in range(5): # i takes values 0, 1, 2, 3, 4
    print(f"Number: {i}")
```

- Iterates over elements of a sequence (lists, strings, ranges, etc.).
- The `range(n)` function generates a sequence of numbers from 0 to n-1.

Functions

```
def greet(name):
    print(f"Hello, {name}!")

def add(a, b):
    return a + b

greet("Alice")
result = add(5, 3)
print(f"The sum is: {result}")
```

- Functions are reusable blocks of code that perform a specific task.
- `def` is used to define a function.
- `return` is used to send back a value.

Lists

```
numbers = [1, 2, 3, 4, 5]
fruits = ["apple", "banana", "cherry"]

print(f"First fruit: {fruits[0]}") # apple
print(f"Last number: {numbers[-1]}") # 5

fruits.append("orange") # Add to the end
print(f"Fruits: {fruits}")

fruits[1] = "blueberry" # Modify an element
print(f"Modified fruits: {fruits}")

print(f"Length of fruits: {len(fruits)}")
```

- Lists are ordered and mutable collections of items.
- They are defined using square brackets `[]`.
- Elements are accessed by index (starting at 0).

Strings

```
greeting = "Hello, "
name = "World"
message = greeting + name + "!" # Concatenation

print(message) # Hello, World!
print(f"Length: {len(message)}")

print(message.upper()) # HELLO, WORLD!
print(message.lower()) # hello, world!
print(message.replace("World", "Python")) # Hello, Python!
```

- Strings are immutable.
- String methods can be used for manipulation (e.g., `upper()`, `lower()`, `replace()`).
- f-strings (`f`) are a convenient way to format strings.

Modules and Packages

```
import math
import random

print(f"PI: {math.pi}")
print(f"Square root of 16: {math.sqrt(16)}")

from random import randint
# Since we imported 'randint' directly, we don't need 'random.'
print(f"Random number between 1 and 10: {randint(1, 10)}")
```

- Modules are Python files (`.py`) that contain functions, classes, and variables.
- They allow you to organize and reuse code.
- `import` is used to include modules.
- `from ... import ...` is used to import specific elements from a module.