Programming

beginner

What is Programming

Programming is giving an order to the computer and it simply executes it.

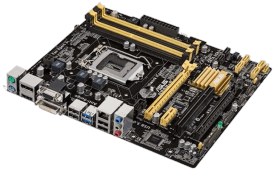
How to Program

Speaking to the computer in a language that it understands.

1

Software + Hardware

Hardware



software

Do you know any other software?

software examples

2

How to create a software?

Step by step

Programming logic

Programming language

Run

3

What is programming logic?

It is the logical reasoning that must be done to solve a problem.

4

What is algorithm?

It is writing in a sequence, logic.

EXAMPLE

Think about the logic of how to bathe.

EXAMPLE BATHING (LOGIC)

Basically it is to enter the shower, turn on the water, regulate the temperature, use the soap, rinse and that's it.

EXEMPLO BATHING (ALGORITHM)

1) Go to the bathroom
2) Take off clothes
3) Open the shower
4) Wet the body
5) Pass soap over body
6) Rinse
7) Dry body
8) Dress in clean clothes
9) Exit the bathroom

EXEMPLO BATHING (ALGORITHM)

1) Is the bath available?
2) Yes, proceed | if not, come back another time. ✖
3) enter the bathroom
4) remove the clothes
5) Put the dirty clothes to wash
6) open the shower
7) Adjust water temperature
8) Adjust the water temperature
9) If the temperature is good, start the bath. If not, adjust the new temperature
10) Mojar the body
11) Pass the jabón by body
12) rinse
13) grab the towel
14) dry body
15) Dress up in clean clothes
16) exit the bathroom

1) enter the bathroom
2) take off the clothes
3) open the shower
4) Mojar the body
5) Pass the jabon in the body
6) flush
7) dry the body
8) wear clean clothes
9) exit the bathroom

EXAMPLE SUM 2 AGES

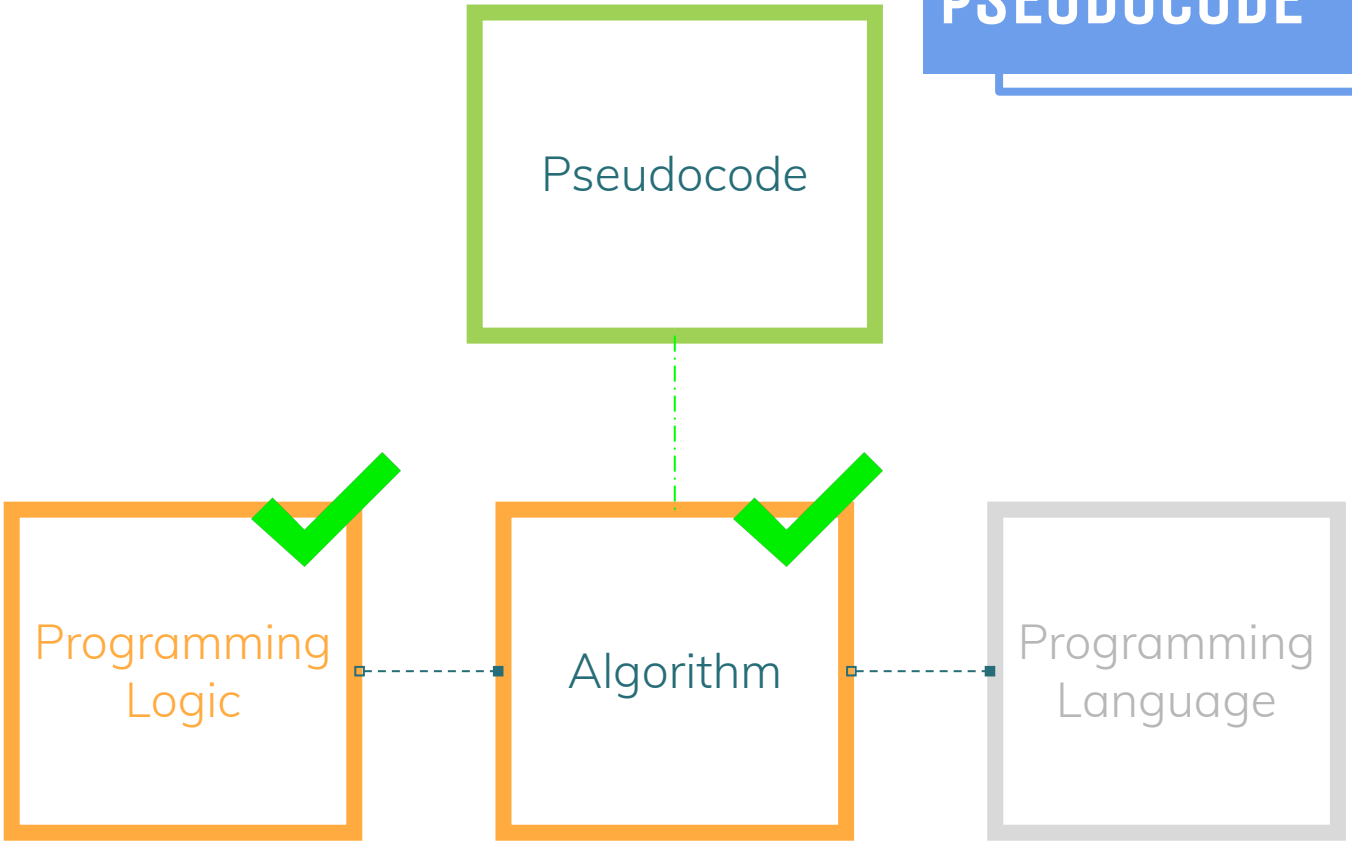
1. read the first age
2. read the second age
3. make the sum of the ages

5

What is pseudocode?

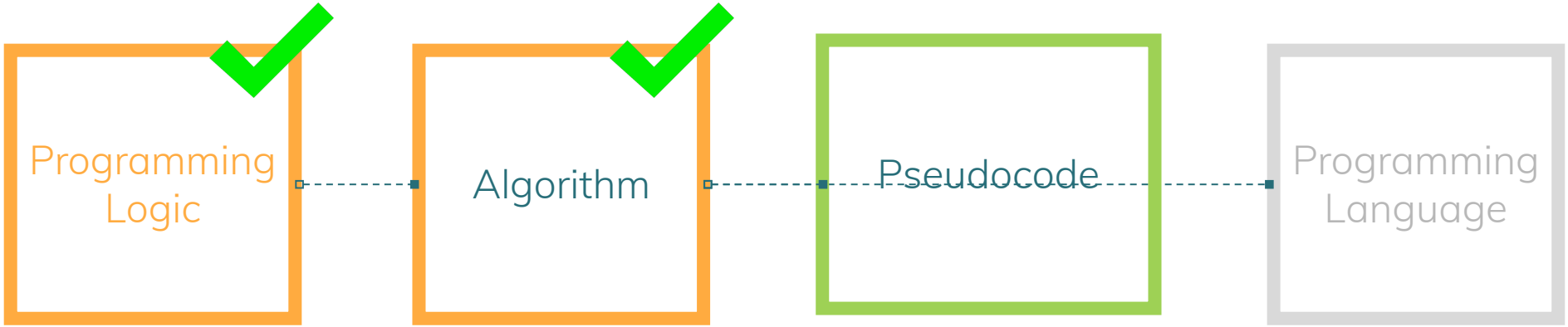
Pseudocode is a generic way of writing an algorithm, using a simple language without the need to know the syntax of any programming language.

PSEUDOCODE


Pseudocode
Programming Logic
Algorithm
Programming Language

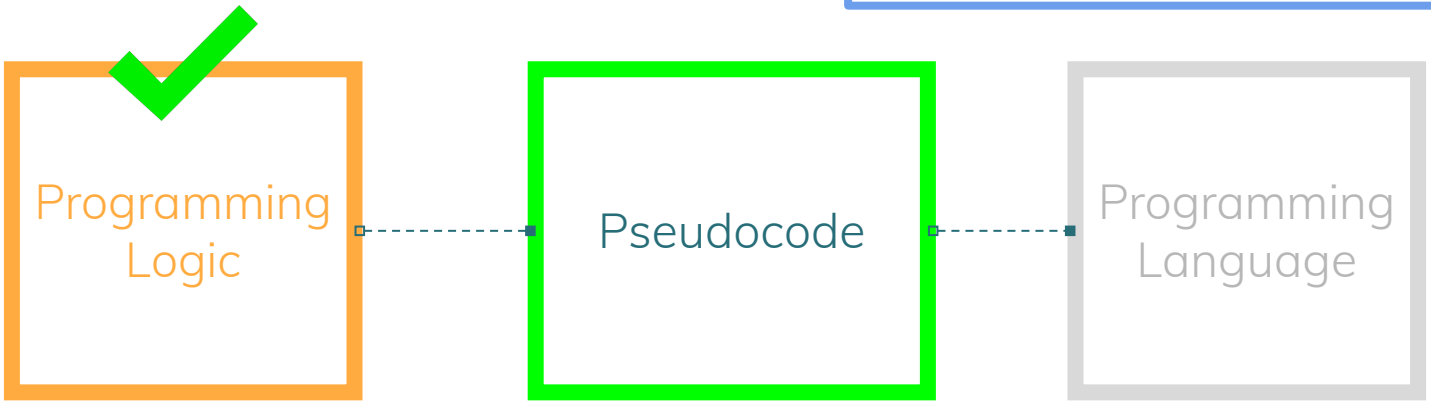
It is not necessary to write the algorithm and then pass it to a pseudocode or programming language.

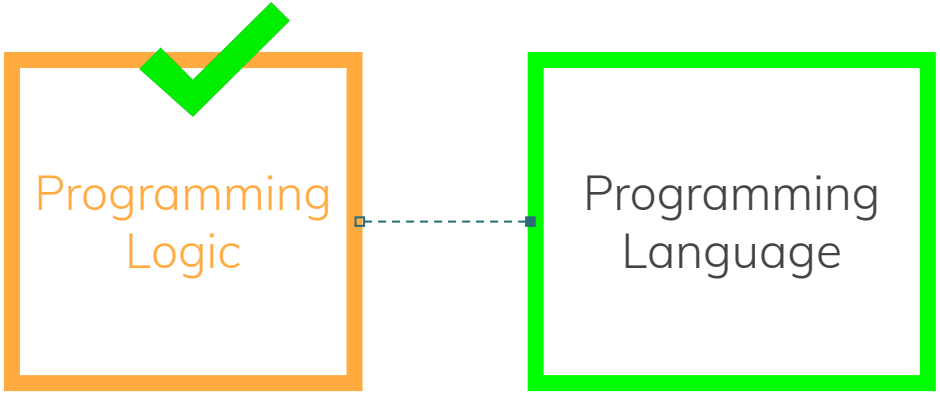
PSEUDOCODE


Programming Logic
Algorithm
Pseudocode
Programming Language

We can think of logic and already write in a pseudo-code or even in a programming language.

PSEUDOCODE


Programming Logic
Pseudocode
Programming Language


Programming Logic
Programming Language

EXAMPLE: ADD 2 AGES

1) read the first age
2) read the second age
3) Create a space in a computer's memory called **result**
4) That space called **result** has to receive

   → Age 1 + Age 2

5) Print the **result** on a screen

EXAMPLE: ADD 2 AGES

Program_sum_of_ages

```
Start

    variables age1, age2, result : integer

    read age1

    read age2

    result ← age1 + age2

    print result

End
```

Declare variables and constants

EXAMPLE: ADD 2 AGES

Program_sum_of_ages

variables age1, age2, result : integer

VARIABLES - because your content may vary!
For example:
age1 can be 1, 2, 50, 100

INTEGER- is the type!
For example:
The age1 must be integer.
There are other types such as decimal, characters(which are letters)

6

Variables x Constants

They are names that we create to represent a value, so that it is possible to use this same value several times in that list of tasks, without having to rewrite every time that is necessary.

VARIABLES AND CONSTANTS

```
program_sum_money

Start

    constants savings_account

    variables current_balance, total

    savings_account= $1000.00

    read current_balance

    total ← savings_account+ current_balance

    print total

End
```

7

Data Types

Example: name, age, salary, gender

For a computer that information is Data.
And the data can take various formats.
Think of it this way, name is different from age which is different from salary.

TYPE 1 : TEXT

Data of type text represents a sequence of one or more characters. They are usually enclosed in double quotes.

An example of a text type:

→ **Name --->** John Connor

→ **Address--->** Regent Avenue

Note: spaces also count as characters. So we have 11 characters in "John Connor".

TYPE 2: INTEGER

They are represented by numerical values, both negative and positive (without decimals).

An example of a given of the integer type:

→ **Age ---> 18**

→ **Number of children---> 5**

TYPE 3 : REAL

Data of type real are numeric values, both positive and negative, that use decimals,

As an example, we can cite:

→ **Salary ---> 13,434.52**

→ **Price ---> 135.70**

TYPE 4 : BOOLEAN

Finally, the logical type. This data type is typically represented by alternatives:

YES or NO, TRUE or FALSE.

Logical data type can also be called Boolean.

An example of logical data is:

→ **Gender--->** male or female

→ **is adult? --->** yes or no

DATA TYPE

Review..

Name: TEXT

Age: INTEGER

Salary: REAL

Gender: BOOLEAN

8

Selection and repetition structure

Organized ways and structures to make a computer capable of making decisions or executing an instruction several times until a pre-established limit is reached, or a condition is satisfied, or the user intervenes.

A computer, different from humans, still cannot think on its own, thus, it needs a well-defined rules for its proper functioning.

This is achieved through a lot of training and practice with programming logic.

Ex: take a bath

1) The bathroom is available?
2) If this continues | if not, come back another hour.
3) enter the bathroom
4) Take out the clothes
5) Put dirty clothes to wash
6) open the shower
7) Regulate water temperature
8) Adjust the water temperature
9) **If the temperature is good, start the bath. If not, adjust the temperature again**
10) wet the body
11) Pass the soap on the body
12) rinse
13) take the towel
14) dry the body
15) wear clean clothes
16) Exit the bathroom

SELECTION AND REPETITION STRUCTURE

Program sum_2_ages

Start

variables age1, age2, result : integer

read **age1**

read **age2**

result ← age1+ age2

print result

End

check if age 1 and age 2 are correct

That is, they are not negative ages.

SELECTION AND REPETITION STRUCTURE

1) read **age1**
2) check if **age1** is correct. if it is correct, go to step 3. If not, ask step 1 again
3) read **age2**
4) Check if **age2** is correct. if it is correct, go to step 5. If not, ask step 4 again
5) Do the sum.

If we get to step 5, we know that all ages were verified, validated.

IF

```
Program_sum_2_ages

Inicio

    variables age1, age2, result : integer

    read age1

    read age2

    IF (age1 > 0) y (age2 >0)

        result ← age1 + age2

        print result

    ELSE

        Print "Please enter correct ages"

    END_IF

END
```

declare variables and constants

You have to validate if they are not negative ages.

WHILE

```
Algorithm_buy_apple

Inicio

    variables my_money, rest : real

    constante apple: real

    leer my_money

    apple = 10.00

    WHILE my_money > apple do:

        my_money ← my_money - apple

        print my_money

    END_WHILE

END
```

declare variables and constants

WHILE

```
WHILE my_money> apple do:
        my_money← my_money← - apple
        print my_money
End_while
```

```
WHILE my_money> apple do:
            rest ← my_money- apple
            my_money← rest
            print  my_money
End_while
```

We put 1 more variable, which would be the rest.

WHILE

```
WHILE my_money> apple do:
        my_money ← my_money - apple
        print my_money
End_while
```

Example:
I have 25 dollars
Apple= 10 dollars

```
WHILE 25 > 10 do
        my_money← 25 - 10
        print 15
```

```
WHILE 25 > 10 do
        my_money← 15 - 10
        print 5
End_while
```

```
WHILE 5 > 10 do
```

WHILE

```
WHILE my_money> manzana do
        rest ← my_money- apple
        my_money← rest
        print my_money
END_while
```

Example:
I have $10
Apple= $10

```
WHILE 28 > 10 do
        rest ← 28 - 10
        my_money←18
        print 18
```

```
WHILE 18 > 10 do
        resto ← 18 - 10
        my_money← 8
        print 8
END_while
```

```
WHILE 8 > 10 do
```

FOR

```
Program buy_apple

Start

    variables my_money, qty: real

    constant apple: real

    read initiol_money

    apple= 10.00

    FOR ( my_money> apple, qty = 0,  qty++)  do

        my_money← my_money- apple

        print my_money

    End_For

    print qty

End
```

qty+=1 or **qty++**

Each time the repetition loop happens, the variable qty adds 1.
Example:
qty+=1
qty++

FOR

```
FOR ( my_money > apple, qty= 0,  qty++)  do
        my_money ← my_money - apple
    print my_money
    End_For
    print  ctd
```

Example:
I have $28
Apple= $10

```
FOR  ( 28 > 10, qty= 0,  qty++)  do
        my_money ← 28 - 10
        print  18
    End_For
    print ctd
```

1

```
FOR  ( 18 > 10, qty= 0,  qty++)  do
        my_money ← 18 - 10
        print  8
    End_For
    print ctd
```

2

```
FOR  ( 8 > 10)  do
```

9

Where can I write?

WHERE CAN I WRITE?

- Notepad
- Code editors (Sublime Text, Vim, Emacs, Visual Studio Code)
- IDE (integrated development environment)(Pycharm , Eclipse, IntelliJ, NetBeans)

10

Programming Languages and Frameworks

PROGRAMMING LANGUAGES

Some languages

- Java
- C
- C++
- Python
- Ruby
- Go (Golang)
- PHP

Frameworks

A set of components that helps build sites faster and easier.

Every language has one or more Frameworks.

PROGRAMMING LANGUAGES

- Java → **Spring, Wizard**
- Python → **Django, Flask**
- Ruby → **Rails, Sinatra**
- PHP → **CakePHP, Laravel**

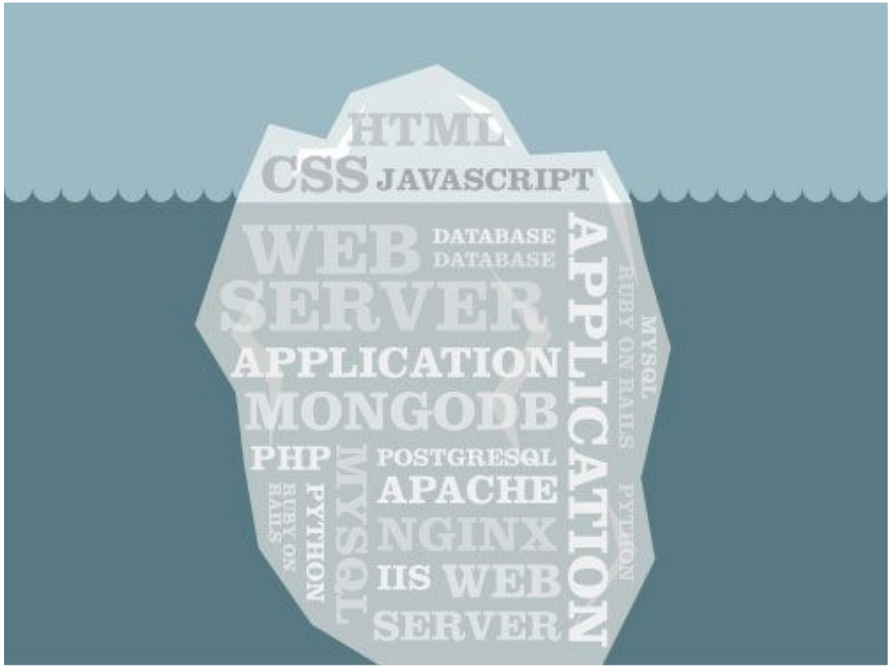
11

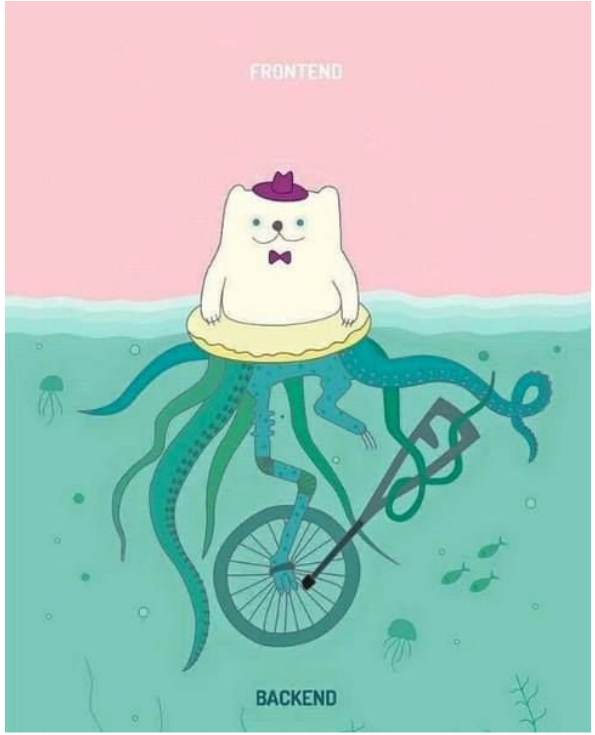
Backend vs Frontend

The frontend programmer is the one who takes care of all the interaction with the user

The backend programmer is the professional who develops the system that will be used for data management, a system that will have interactivity with the user and will use the interface that was developed by the frontend programmer.

BACKEND VS FRONTEND





THANKS