Unit 4. Basic Programming

Héctor D. Menéndez

Endless Science

endsci.net

Index

Index

Programming

What is Programming?

Computer programming (or programming) is the comprehensive process that leads from an original formulation of a computing problem to an executable program. It involves:

- The analysis.
- The abstraction.
- The interpretation of problems in algorithms.
- Verification of requirements including its correctness and its resource optimization.
- The coding of the algorithm in a programming language.
- Test, debug, maintain and reuse the code.

Programming

What is Programming?

It can be considered as a balanced measure among:

- An artistic or creative process.
- A craft.
- An engineering discipline.

Software Quality

A program quality is measured by:

Reliability: The results should be correct. This depends on conceptual correctness of algorithms, and minimization of programming mistakes, such as mistakes in resource management and logic errors (e.g. division by zero).

Robustness: The program's ability to anticipate problems that are not part of the program's logic.

Usability: The ability of a program interface to be intuitive for the final users.

Software Quality

Portability: The range of computer hardware and operating system platforms on which the source code of a program can be compiled/interpreted and run.

Maintainability: The degree of understanding of a software to be easily modified by its present or future developers to make improvements or customizations, fix bugs and security holes, or adapt it to new environments.

Efficiency: the amount of system resources a program consumes.

Index

Program

What is a Program?

A computer program, or just a program, is a sequence of instructions, written to perform a specified task within a computer.

Program Design

Design is the process which establishes the requirements, goals, algorithms and different logic and abstraction levels of a program.

- **Abstraction**: It is the conceptual idea of the program and every program part.
- **Modularity**: The different components which conform the program.
- **Flow**: Program behaviour for every action.
- **Data Structure**: The structure of the processed data within the program.
- **Software Architecture**: Final program construction which links all the modules through the flow to establish the general program logic.

Program Design: Programming Paradigm

A programming paradigm is a fundamental style of computer programming, a way of building the structure and elements of computer programs. Some paradigms are:

- **Structured**: This paradigm uses only subroutines and three kind of structures: sequences, selections and repetitions. It tries to avoid unconditional jumps (Goto instruction).
- **Object Oriented**: Concepts are represented as objects which have attributes and methods. The interactions among the objects conform the program.
- **Functional**: This paradigm uses successive function calls recursively.

Index

Program Design: Example

Definition

Create a program that manages tasks. The program allows to add or remove information about the tasks.

Abstraction: The program should:

- Add data about tasks.
- Remove data about tasks.

A single list will be used to add or remove tasks.

Program Design: Example

Definition

Create a program that manages tasks. The program allows to add or remove information about the tasks.

Modules: The different modules are:

- **List**: Allows to add o remove elements.
- **User**: It communicates with the user.
- **Data**: Generates the data structure.
- **Principal**: It manages the modules and the I/O interface.

Program Design: Example

Definition

Create a program that manages tasks. The program allows to add or remove information about the tasks.

Flow: The users runs the program and she has to choose among:

- Add a new task.
- Remove a task.
- Show tasks.
- Exit.

Program Design: Example

Definition

Create a program that manages tasks. The program allows to add or remove information about the tasks.

Flow: Add a task.

- Ask the user the task that she wants to add.
- Add the task into the list.
- Return to the beginning.

Program Design: Example

Definition

Create a program that manages tasks. The program allows to add or remove information about the tasks.

Flow: Remove a task.

- Ask the user about the task that she wants to remove.
- Remove the task from the list.
- Return to the beginning.

Program Design: Example

Definition

Create a program that manages tasks. The program allows to add or remove information about the tasks.

Flow: Show tasks.

- Get the whole list of tasks.
- Print it.
- Return to the beginning.

Program Design: Example

Definition

Create a program that manages tasks. The program allows to add or remove information about the tasks.

Flow: Exit.

- Free the resources and close the program.

Program Design: Example

Definition

Create a program that manages tasks. The program allows to add or remove information about the tasks.

Data structure: The data structure will use the following information:

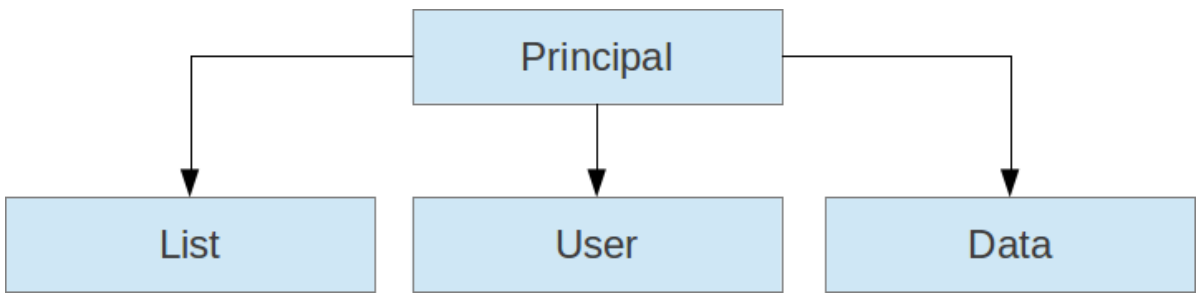
- Id: identification number for each task.
- Person name.
- Task name.
- Task deadline.

Program Design: Example

Definition

Create a program that manages tasks. The program allows to add or remove information about the tasks.

Software Architecture: Several architectures can be chosen.



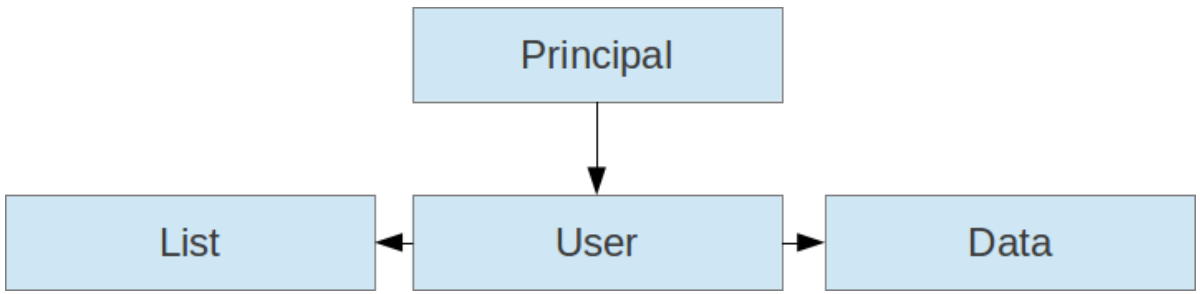
- List functionalities: Adds or removes task.
- User functionalities: General Menu, Ask for data (tasks), Ask for Questions (exit?).

Program Design: Example

Definition

Create a program that manages tasks. The program allows to add or remove information about the tasks.

Software Architecture: Several architectures can be chosen.

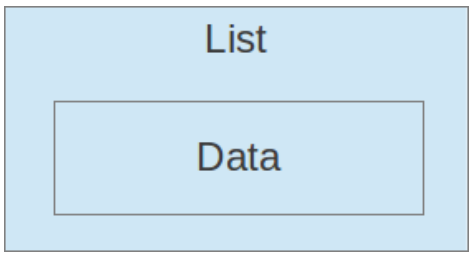


Program Design: Example

Definition

Create a program that manages tasks. The program allows to add or remove information about the tasks.

Software Architecture: Several architectures can be chosen.



Index

Coding

What is coding?

Process which translates the design to a source code through a Programming language.

Coding

Algorithm: Sequence of actions used to solve a problem.

Function: code fraction which runs a specific task.

Source code: Written by the developer.

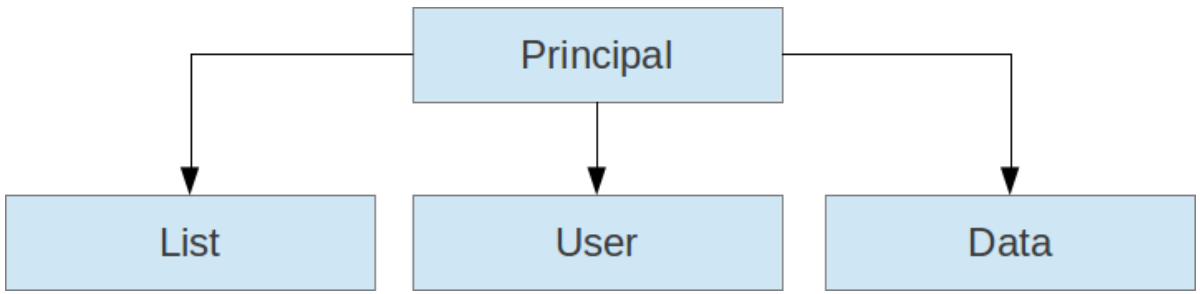
Libraries: External functions used for a specific task.

Compiler: Generates the binary files from the source code.

Interpreter: Run the source code line by line.

Executable: binary file which is runnable.

From Code to Binary



Principal: Use I/O libraries.

List and data: libraries.

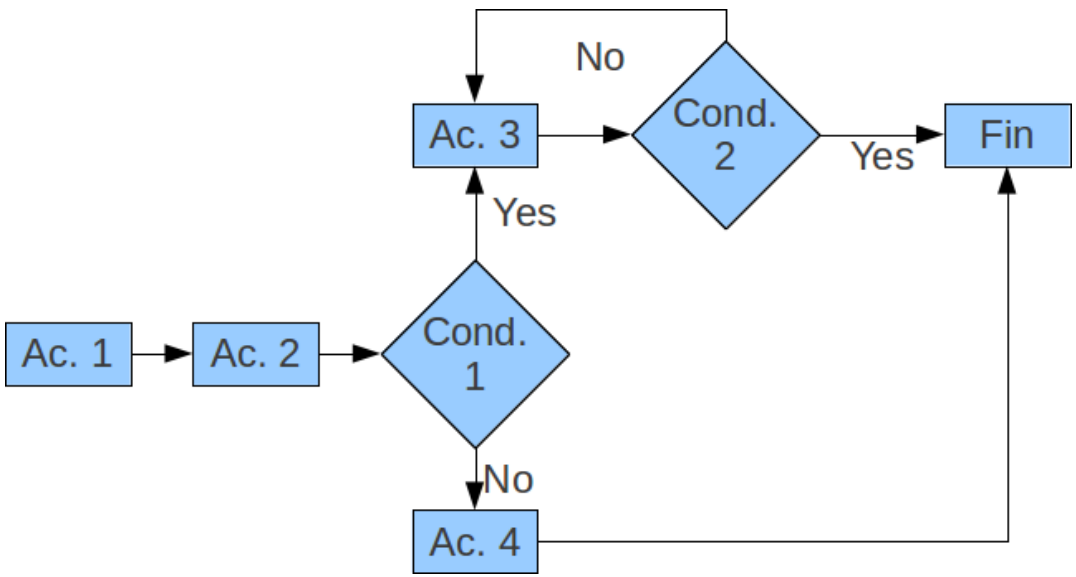
User and Principal: Source code.

Functions: general menu, ask data, ask user...

The paradigms in coding

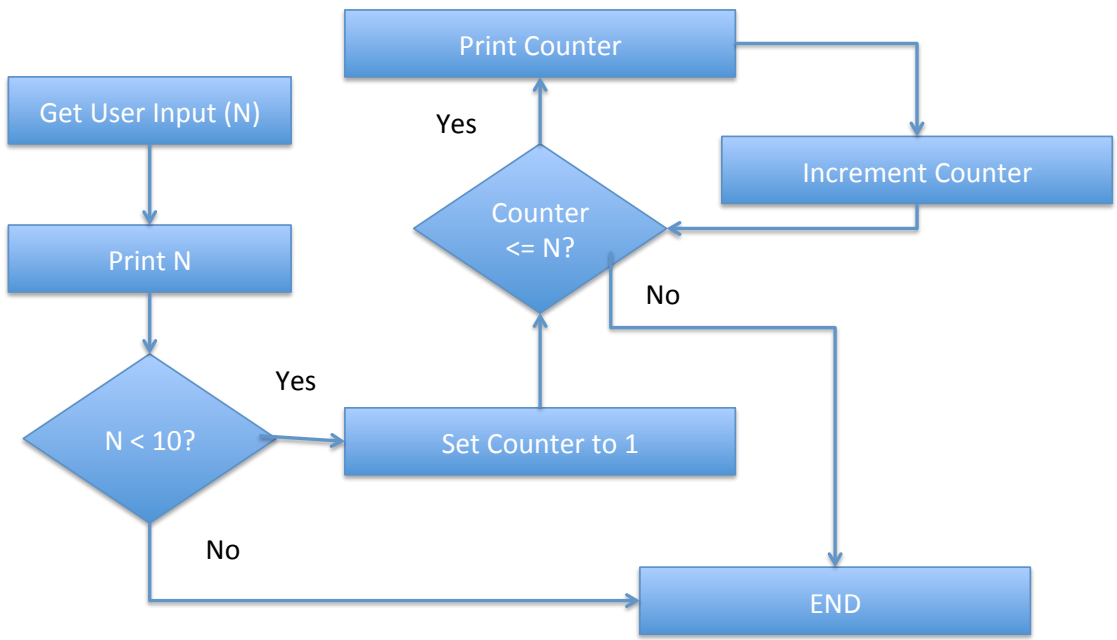
During this course we will use two paradigms: structure and object-oriented programming. Every program will be written with three kind of instruction:

- Sequential.
- Selective.
- Repetitive.



The paradigms in coding

Imagine a program that ask the user for a number and prints every number from one to the user's number but only if the number introduced by the user is less than 10. The flow would be:



First Program

Open by typing `python3` in your terminal.

Write the code: `print(“Hello World”)`.

```
$> python3
>>> print("Hello World")
>>> quit()
```

Listing 1: Opening the Interpreter

The program will simply print "Hello World". To close the interpreter use `quit()`.

First Script

Create a file called script.py (or choose any name you want).

Write the code: `print(“Hello World”)` inside of the file.

Run the file by typing `python3 script.py` in your terminal.

```
$> nano script.py
$> python3 script.py
```

Listing 2: Opening the Interpreter

The program will simply print "Hello World".

Index

Exercise 1: Program Design

Imagine that you have to design the WhatsApp or Telegram app.
Try to think on the previous concepts to propose a high level design.
You can abstract servers and databases as external resources.

What would be the abstraction?

Which different modules would you consider?

What would be the flow of the program?

What data structure would you use?

Exercise 2: Flow Design

Imagine a program that draws a triangle with * symbols. The triangle will have N levels, where N is provided by the user. Create a program flow that draws the triangle by levels starting at the top and follows the pattern:

```
N=2          N=3           N=4
 *            *             *
***          ***           ***
            *****         *****
                         *******
```

The flow can contain a counter that starts in 1 and can be incremented by one. There are also two special instructions "draw #number spaces" or "draw #number *", which will print as many spaces and * as consider. Use N and counter to decide these numbers.