college CDI

Fundamental notions of programming (NFP)

Class 2

Pseudocode

We have seen that it is necessary to analyze and plan any programming project. Because the computer needs to be told exactly what to do in every possible circumstance, we must decompose the entire program into an algorithm stating all the inputs and instructions in order to reach given goals.

Although pseudocode isn't mandatory, it is a great way to plan programming and an excellent way to become good at programming faster.

The first step into programming is to write a pseudo code which is pretty much what you have been doing for assignment 1. Not using any programming language, you have described what was needed to do in order to reach you objective. Of course, knowing the possibilities offered by different programming language helps you writing an efficient and useful pseudocode.

But we oversimplified the processes so you can understand the overall logic. Although, computers need much more details.

Example : taking a glass of water at night
Imagine you wake up thirsty at night and you want to drink a glass of water. Let's write a program for this the way we did :

- Get out of bed
- Go to the kitchen
- Get a glass from the cupboard
- Turn on the faucet
- Fill the glass
- Drink the water from the glass

This is quite a good resume of the steps needed to accomplish the task. But the instructions aren't detailed enough for the computer. The computer would more need something like the following :

- While unfolding your knee and folding the right leg
    - Move your left leg forwards 12 inches
- Repeat same process with the opposite side to the fridge
- Put your left hand on the fridge's door
- Pull until door reaches 45 degrees
- Look for the bottle of water
- While opening your hand, reach for the bottle
- Etc.

This is much more precise as literally everything needs to be told to the computer. If the computer would be a living creature, the program would need to tell its heart to beat every second.

How to write pseudocode

Pseudocode writing is a subjective and non-standard process which doesn't take consideration of precise programming languages. Here are a few guidelines to help you write them:

Resume the purpose of the program
To start with, resume efficiently the purpose of the program you want to create.

One instruction per line
Each line expresses one (and only one) instruction given to the computer.

Use capitals for key commands
Capitalize key commands to highlight their importance (e.g.: IF, PRINT, etc.).

Focus on the actions
Write what the program has to do, not how it should do it.

Use standard programming structures
Follow the algorithm flow as we have introduced it and create an overall programming structure which is as clear and simple as possible.

Comment your pseudocode
Whenever necessary, use comment to clearly state what's needed

Create blocks of instructions
As much as possible, group similar actions into blocks to clearly separate the program in several steps using white space and indentation.

Pseudocode examples

Print a message whether a student passed or failed a class

Resume:
The program requests a student mark. If the mark is over 60, the message «Passed» is printed, and if the mark is under 60, the message «Failed» is printed.

```
BEGIN
        USER ENTERS student mark

        IF student's grade is greater than or equal to 60
                Print "passed"
        ELSE
                Print "failed"
END
```

Output a greeting*

Resume:
This program will request a greeting from the user. If the greeting matches a specific response, the response will be delivered; if not, a rejection will be delivered.

```
BEGIN
        PRINT greeting                          // Greeting
                "Hello stranger!"
        PRINT prompt
                "press «Enter» to continue"
        <user presses "Enter">

        PRINT call-to-action                    // Call-to-action
                "How are you today?"

        DISPLAY possible responses
                "1. Fine."
                "2. Great!"
                "3. Not good."

        PROMPT                                  // User input
                "Enter the number that best describes you:"

        IF "1"
                PRINT response
                        "Dandy!"
        IF "2"
                PRINT response
                        "Fantastic!"
        IF "3"
                PRINT response
                        "Lighten up, buttercup!"

        ELSE                                    // Error message
                PRINT response
                        "Enter a valid choice, please."
END
```

*Example taken from wikihow.com

Flowchart

Sometimes called process flowchart, process map or process flow diagram (PFD), the purpose of a flowchart is to represent an algorithm in a more functional way. They can consist into simple, hand-drawn diagrams or into very complex computer-drawn charts representing various possible steps and routes. Representing the data flow, flowcharts are useful to fragment an algorithm and to explain it to others. It is often created based on the pseudocode before starting the programming work.

How to create a flowchart

Flowcharts uses different containers, symbols, lines and arrows. Here are some of the most commonly used.

Terminal
Rounded rectangles (terminal points) indicate the flowchart's starting and ending points.

Flow Lines
Default flow is *left to right* and *top to bottom*. Arrowheads are often only drawn when the flow lines go contrary the normal.

Input/Output
Parallelograms represent input or output operations.

Process
Rectangles represent processes (e.g.: mathematical computation, variable assignment, etc.).

Decision
Diamonds represent the *true/false* statement being verified.

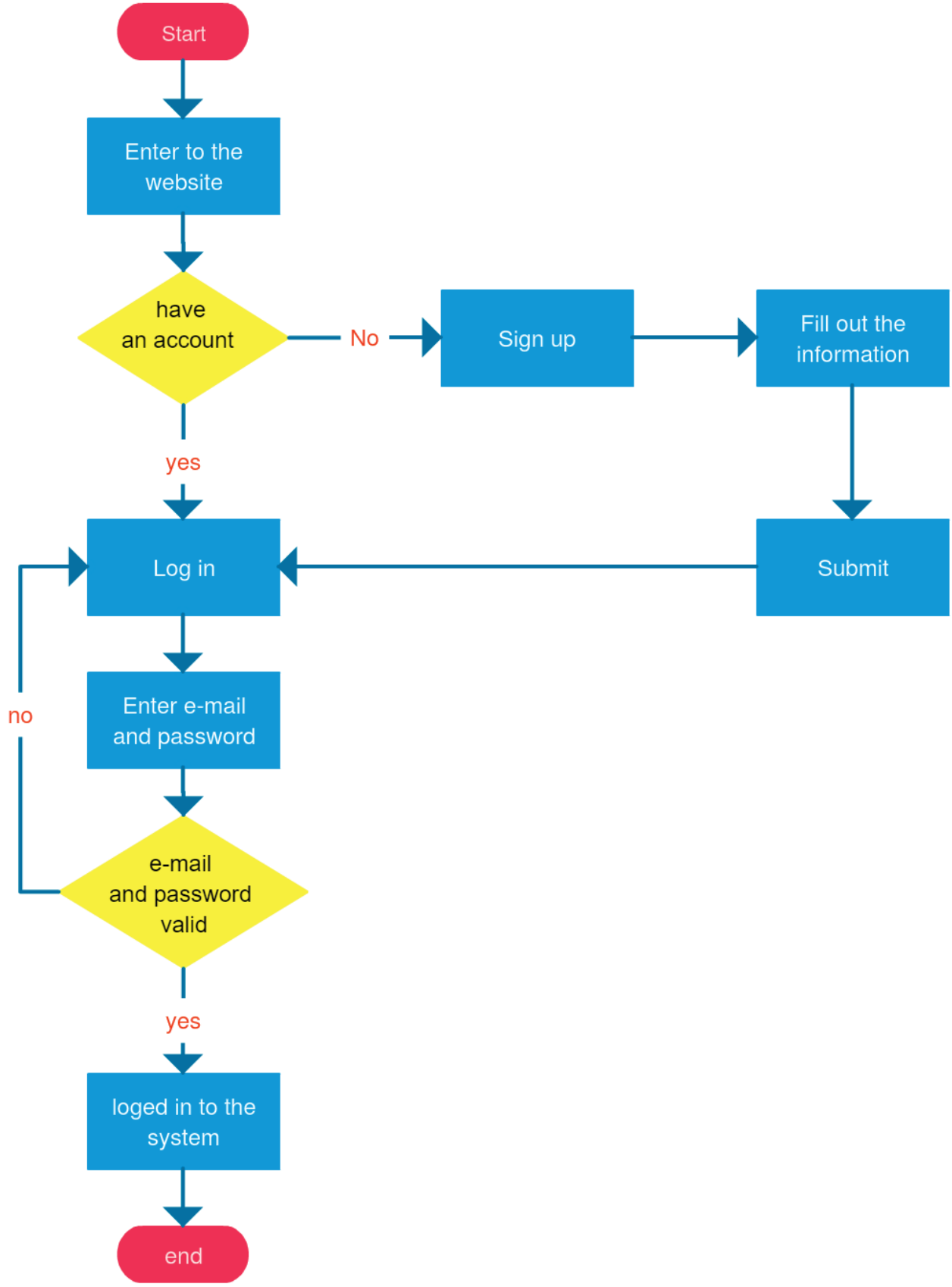
On-page Connector
Circle is used to join different flow lines.

Off-page Connector
Used to connect flowchart portion on different page.

Module Call / Predefined Process/Function
Used to represent a group of statements performing one processing task.

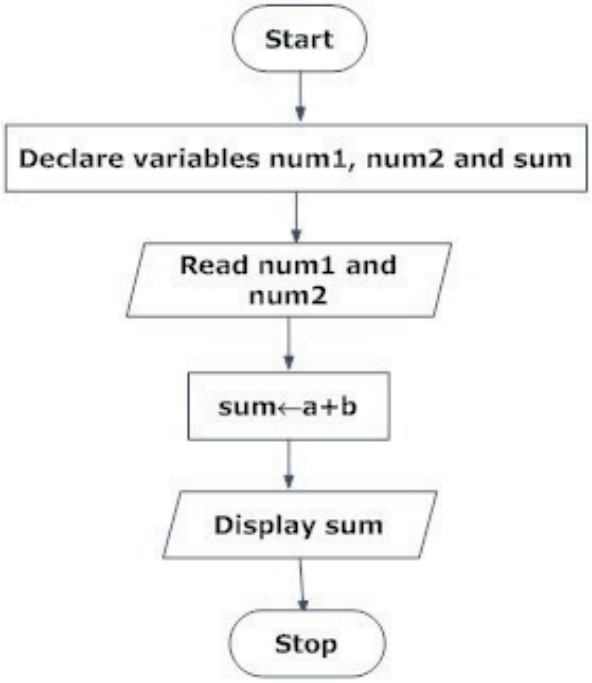
Often programmers make a distinction between program control and specific task modules or between local functions and library functions.

Flowchart example

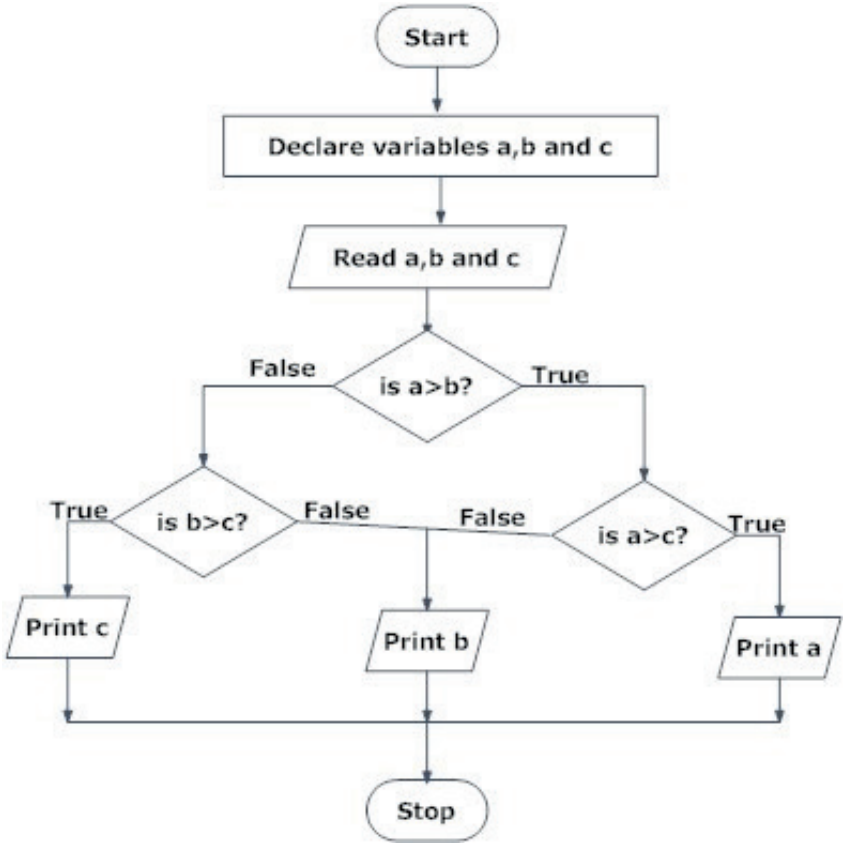


Examples

Flowchart to add two numbers entered by user.



Flowchart to find the largest among three different numbers entered by user.



Assignment 2 :
Create the pseudocode and flowchart

The program will ask for the user to write 10 different numbers between and including 1 and 100.

The program will print the user's chosen numbers, will calculate and print the average of the 10 numbers as well as the sum of the numbers.