Pseudocode and Flow Chart

- Concept of Pseudocode and Flow Chart
- Apply the concept on our daily lives

When programmers plan the logic for a solution to a programming problem, they often use one of two tools

> Pseudocode ("sue-doe-code")

> flowcharts

Pseudocode

- is an English-like representation of the logical steps it takes to solve a problem.

flowchart

- is a pictorial representation of the same thing.

Pseudocode

- **Pseudo** is a prefix that means “false,”
- **code** a program means to put it in a programming language;
- therefore, **pseudocode** simply means “false code,” or sentences that appear to have been written in a computer programming language but do not necessarily follow all the syntax rules of any specific language.

Pseudocode

Number Doubling Problem:

```
start
    input myNumber
    set myAnswer = myNumber * 2
    output myAnswer
stop
```

Using pseudocode involves writing down all the steps you will use in a program.

Pseudocode

Pseudocode is fairly flexible because it is a planning tool, and not the final product. Therefore, for example, you might prefer any of the following:

- start and stop == begin and end
- input myNumber ==get myNumber == read myNumber
- set myAnswer = myNumber * 2 == calculate myAnswer = myNumber times 2 == compute myAnswer as myNumber doubled
- output myAnswer == display myAnswer == print myAnswer == write myAnswer.

Pseudocode

Pseudocode statements are instructions to retrieve an original number from an input device and store it in memory where it can be used in a calculation, and then to get the calculated answer from memory and send it to an output device so a person can see it.

When you eventually convert your pseudocode to a specific programming language, you do not have such flexibility because specific syntax will be required.

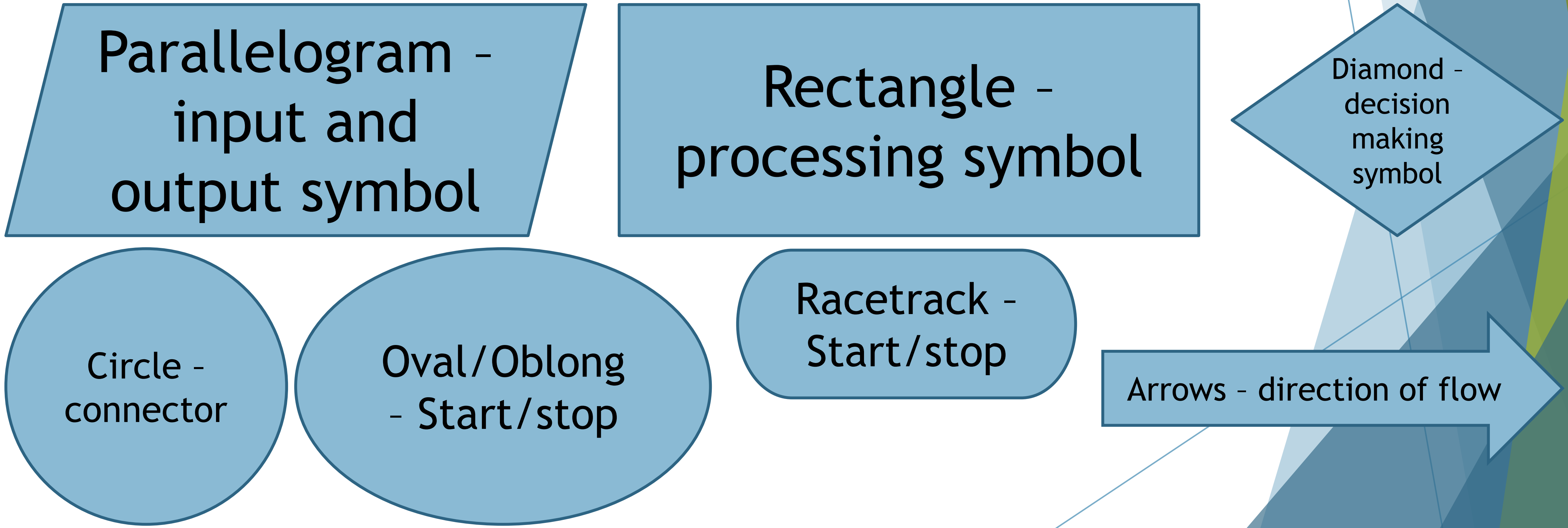
Using pseudocode is more similar to writing the final statements in the programming language.

Flowcharts

- Flowcharts are an excellent tool to help them visualize how the statements in a program are interrelated.
- Using flowcharts allow programmers to visualize more easily how the program statements will connect.
- When you create a flowchart, you draw geometric shapes that contain the individual statements and that are connected with arrows.

Flowcharts

The geometric shapes that contain the individual statements and that are connected with arrows are the following:


Parallelogram – input and output symbol
Rectangle – processing symbol
Diamond – decision making symbol
Circle – connector
Oval/Oblong – Start/stop
Racetrack – Start/stop
Arrows – direction of flow

Flowcharts

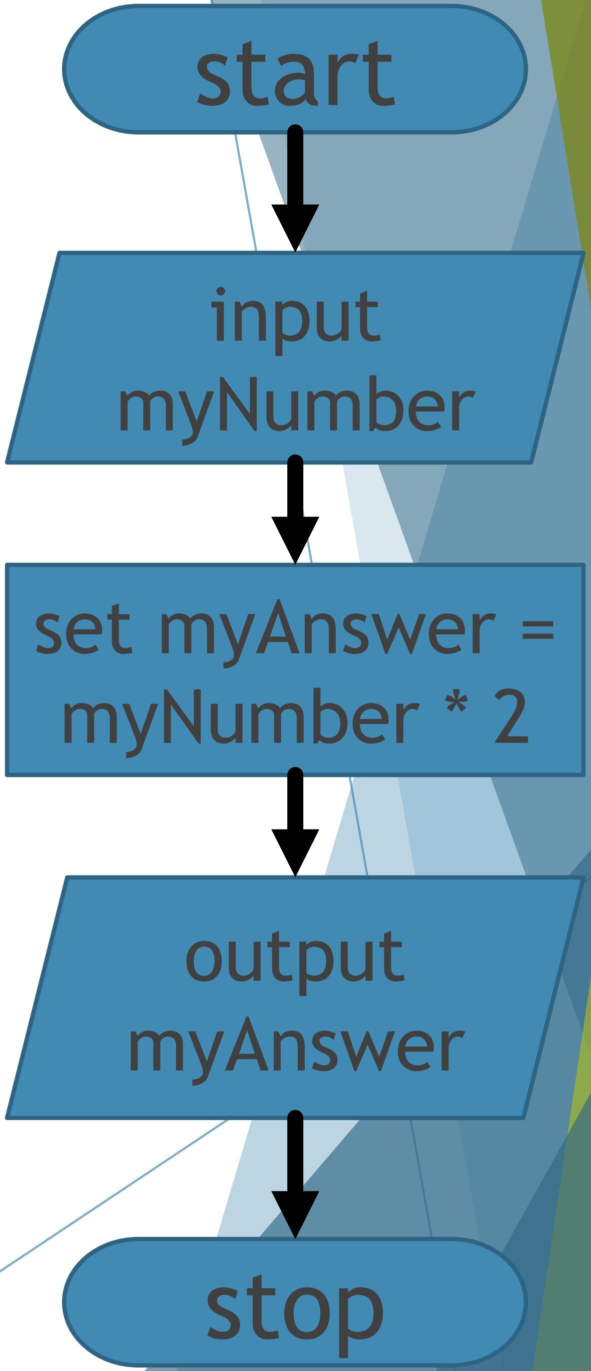
- To show the correct sequence of these statements, you use arrows, or **flowlines**, to connect the steps. Whenever possible, most of a flowchart should read from top to bottom or from left to right on a page. That’s the way we read English, so when flowcharts follow this convention, they are easier for us to understand.

Flowcharts

- To be complete, a flowchart should include two more elements: **terminal symbols**, or start/stop symbols, at each end. Often, you place a word like start or begin in the first terminal symbol and a word like end or stop in the other.
- The standard terminal symbol is shaped like a racetrack; many programmers refer to this shape as a lozenge, because it resembles the shape of the medication you might use to soothe a sore throat.

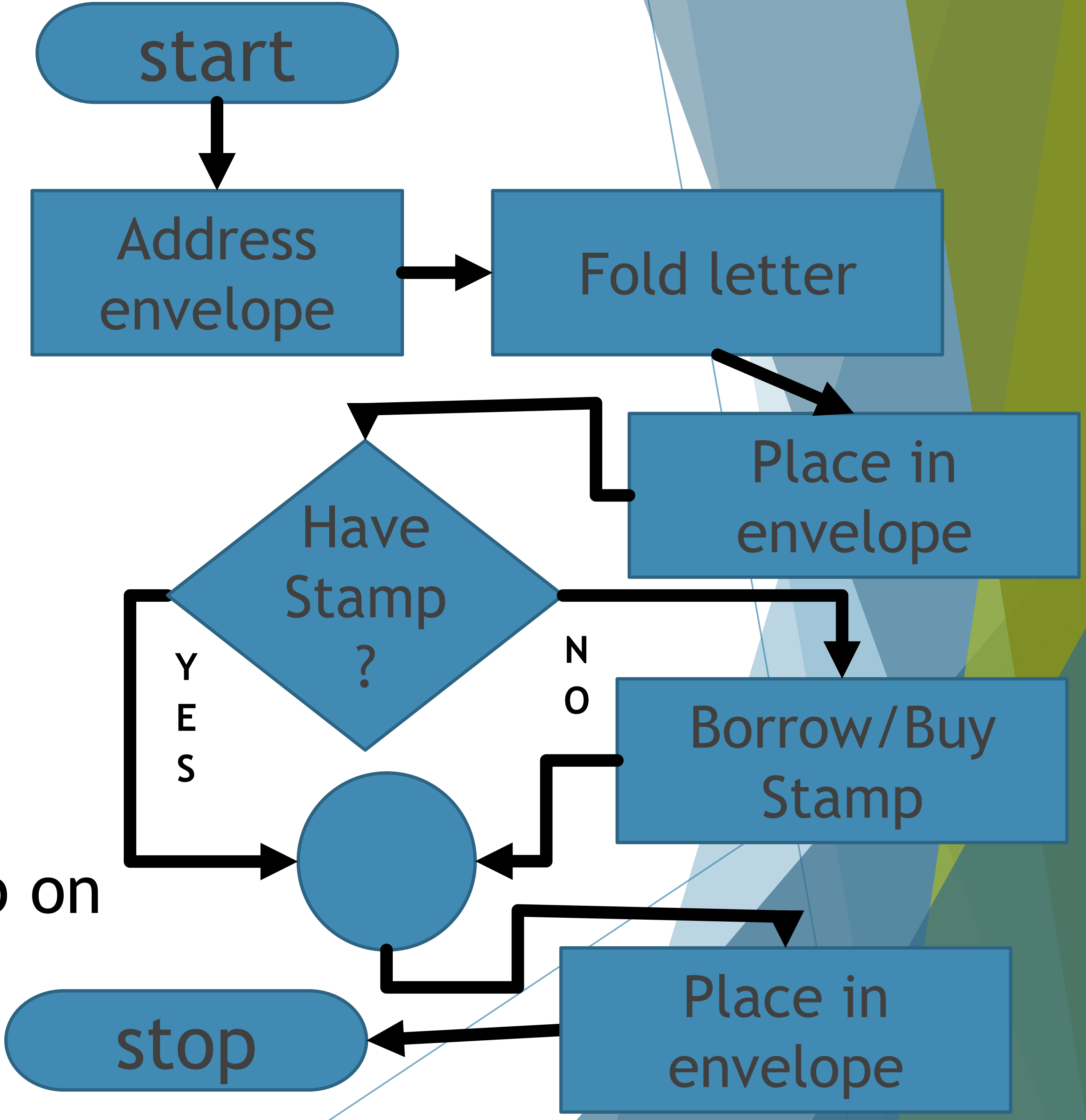
Number Doubling Problem:

```
start
    input myNumber
    set myAnswer = myNumber * 2
    output myAnswer
stop
```



Letter to Envelope Problem:

```
start
        Address envelope
        Fold letter
        Place in envelope
        Have Stamp?
        Yes - connector
        No – Borrow/Buy Stamp
        Connector – Place stamp on
        envelope
stop
```



Any questions???

Applications???

REFERENCES:

- Farrell, J. (2011). *Programming Logic and Design Comprehensive. Sixth Edition.* *https://drive.uqu.edu.sa/_/fbshareef/files/farrell23936_1111823936_02_01_chapter01.pdf*
- *Programming Logic and Design Comprehensive. Sixth Edition.* *https://websites.delta.edu/donaldsouthwell/cst170/ch01_ppt.pdf*
- Computer Programming. (n.d.). https://homepage.cs.uri.edu/faculty/wolfe/book/Readings/Reading13.htm

```
Start
    Input coffee
    input hot water
    input sugar
    stir the ingredients
    taste the coffee
    is the coffee sweet or not?
    Yes –connector
    No- input sugar
    Connector – drink your coffee
stop
```