Problem Solving & Algorithms

UNIMASMOOC
UNIVERSITI MALAYSIA SARAWAK

OVERVIEW

- What is Problem Solving?
- How to solve a problem?
- Problem Solving Strategies
- Design & representation of algorithms
  - (1) Pseudocode and (2) Flow chart

LEARNING OUTCOME

At the end of this unit, students should be able to:

- Define what is problem solving.
- Discuss different problem strategies.
- Apply suitable tools to solve a given problem.

What is Problem Solving?

- **Definition:**
  - **Problem solving** is the **process of transforming** the description of **a problem** into the **solution** of that problem by **using our knowledge** of the problem domain and by relying on our ability to **select** and **use appropriate** problem solving **strategies, techniques and tools**.

Problem Faced in Everyday in Life

We make decisions everyday

Examples:

- Should I wear casual or formal today?
- Should I watch TV or go out to cinema?
- Where is my c programming lab?
- Where to put my bag?

Everything needs a **DECISION AS A SOLUTION TO THE PROBLEM**

Steps to solve a problem

- **Step 1**: **Identify** the problem
- **Step 2**: **Determine** the problem solving strategy to apply
- **Step 3**: **Design** the solution
- **Step 4**: **Execute** the solution
- **Step 5**: **Evaluate** the successfulness of the solution

Problem Solving Strategies

- Many strategies can be use in solving problem.
- Most popular type of strategy is the “Art of War” by Sung Zhi (Ancient Chinese Philosopher)
- We will discuss the following strategies:
    1. Guess and Check (Try an Error)
    2. Start at the end (Work Backward)
    3. Divide and Conquer
    4. Look for a Pattern

1. Guess and Check (Try an Error)

- Simplest strategy available.
- Just experimenting with the possible solution.
- A bit time consuming if have a lot of possible solution
- But it is very efficient and practical for solving small or medium problem
- Example

  Ahmad divided 15 stone games into two piles: games he owns and games his brother owns. He owns 3 more games than his brother. How many games does his brother own?

- Sample Solution
  - I'll guess his brother owns 8 games.
  - That means Ahmad owns 11 games. That's a total of 19 games.
  - My guess is too high.
  - I'll guess again. This time I'll guess his brother owns 6 games.
  - That means Ahmad owns 9 games. That's a total of 15 games.
  - My guess is right.
  - His brother owns 6 games.

2. Start at The End (Working Backward)

- A problem may tell you what happened at the end of a series of steps and ask you to find what happened at the beginning.
- To solve the problem, work backward step by step to the beginning.
- Also known as bottom-up approach

Remember:

The answer is found by starting with the end result and working back to the beginning.

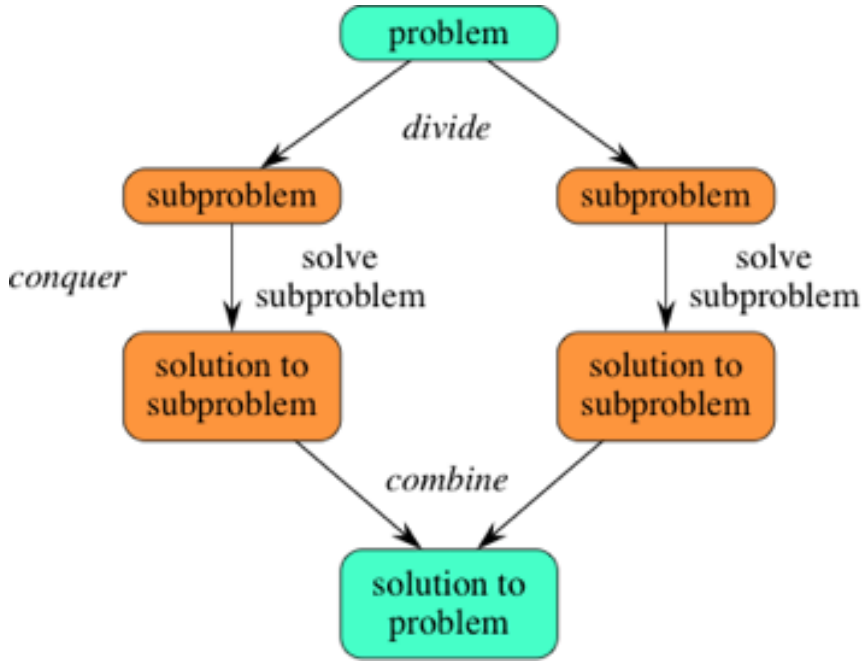
Example:

The waiter brought in 4 pies left over from the dinner. 12 pies were eaten at the dinner. Ahmad took 2 home with him. How many pies did the waiter bring into the feast at the beginning?

- Solution
  - First, I'll account for all the pies that were eaten or taken home.
  - 12 + 2 = 14
  - Then I'll add the 4 pies that were left over.
  - 14 + 4 = 18
  - Therefore, there must have been 18 pies at the start of the dinner.

3. Divide and Conquer

- Most popular strategy used in problem solving.
- Used by Napoleon in his quest to conquer the world.
- Often break up large problems into smaller units that are easier to handle.



4. Look for a pattern

- More advance way to solve a complicated problem
- Some problem can be solved by recognizing pattern within the domain.
- For example:

Johan arranged tray of **KEK LAPIS** on 6 shelves in the shop. He put 1 trays on the top shelf, 3 trays on the second shelf, and 5 trays on the third shelf. If he continues this pattern, how many trays did Johan put on the 6th shelf?

- Solution

Shelf	1	2	3	4	5	6
TRAY	1	3	5	7	9	11

Other Strategies

- Drawings/modeling
- Logical Reasoning
- Extra Information

TIPS:

STRATEGY IS ONLY TO HELP YOU TO SOLVE PROBLEM. MOST IMPORTANT THING IS KEEP YOUR MIND COOL AND TRY TO FIND WAYS TO OVERCOME THE DIFFICULTIES!!

ALGORITHM

UNIMASMOOC
UNIVERSITI MALAYSIA SARAWAK

What is an algorithm?

- Designing software usually is designing an ***Algorithm.***
- **An Algorithm** is a **sequence** of a **finite number** of steps arranged in a specific logical order, which, when executed, produce the solution for a problem.

Algorithm Requirements

- An algorithm must satisfy some requirements:
  1. **Unambiguousness**
     - It must not be ambiguous.
     - Computers cannot cope with ambiguous.
     - Therefore, every step in an algorithm must be clear as to what it is supposed to do and how many times it is expected to be executed.
  2. **Generality**
     - It must have generality.
     - A procedure that prints the message ***"One inch is 2.54 cm"*** is **not** an algorithm;
     - however, one that converts a supplied number of inches to centimeters is an algorithm.

Algorithm Requirements

3. Correctness

– It must be correct and must solve the problem for which it is designed.

4. Finiteness

– It must execute its steps and terminate in finite time.

– An algorithm that never terminates is unacceptable.

Something to ponder …

What is the connection between the real life processes and algorithm?

Algorithm in Real Life

Consider the following ….

Problem: Baking a Cake

How to solve:

1. Start
2. Preheat the oven at 180^{o}C
3. Prepare a baking pan
4. Beat butter with sugar
5. Mix them with flour, eggs and essence vanilla
6. Pour the dough into the baking pan
7. Put the pan into the oven
8. End

Representing Algorithm

- A specific and step-by-step set of instructions for carrying out a procedure or solving a problem, usually with the requirement that the procedure terminate at some point

- 2 types of algorithm representation will be discussed:
  1. **Flowchart**
  2. **Pseudocode**

Problem: Prepare a Breakfast

1. Start
2. **Prepare a Breakfast**
3. End

What is a Pseudocode?

- a semiformal, English-like language (or other language that can be understand by human being)
- limited vocabulary that can be used to design and describe algorithms.
- Not executed on computers
- A carefully prepared pseudocode program may be converted easily to a corresponding C program

Pseudocode

- A pseudocode can be used for:
  - Designing algorithms
  - Communicating algorithms to users
  - Implementing algorithms as programs
  - Debugging logic errors in program
  - Documenting programs for future maintenance and expansion purposes

- A pseudocode must meet these requirements:
  - Have a limited vocabulary
  - Be easy to learn
  - Produce simple, English-like narrative notation
  - Be capable of describing all algorithms, regardless of their complexity

How to write Pseudocode?

- An algorithm can be written in pseudocode using six (6) basic computer operations:

1. **A computer can receive information.**
2. **A computer can output (print) information.**
3. **A computer can perform arithmetic operation**
4. **A computer can assign a value to a piece of data:**
5. **A computer can compare two (2) pieces of information and select one of two alternative actions.**
6. **A computer can repeat a group of actions.**

How to write Pseudocode?

1. **A computer can receive information.**

   Typical pseudocode instructions to receive information are:

```
Read name
Get name
Read number1, number2
```

How to write Pseudocode?

2. **A computer can output (print) information.**

   Typical pseudocode instructions are:

```
Print name
Write "The average is", ave
```

How to write Pseudocode?

3. **A computer can perform arithmetic operation**

   Typical pseudocode instructions:

```
Add number to total
Total = Total + Number
Ave = sum/total
```

How to write Pseudocode?

4. **A computer can assign a value to a piece of data:**

To assign/give data an initial value:

```
Initialize total to zero
Set count to 0
```

To assign a computed value:

```
Total = Price + Tax
```

How to write Pseudocode?

5. **A computer can compare two (2) pieces of information and select one of two alternative actions.**

   Typical pseudocode e.g.

```
If number < 0 then
   add 1 to neg_number
else
   add one to positive number
end-if
```

How to write Pseudocode?

6. **A computer can repeat a group of actions.**

Typical pseudocode e.g.

```
Repeat until total = 50
   read number
   write number
   add 1 to total
end-repeat
```

OR

```
while total < = 50 do:
   read number
   write number
end-while
```

Example: Sum of 2 numbers

```
Start
Read status
if
  status is equal to 1
        print “on”
Else
If
  status is equal to 0
    print “ Off”
else
  print “Error in status code”
end_if

End
```

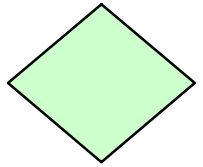
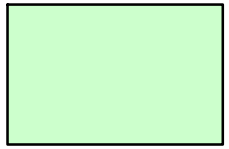
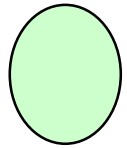
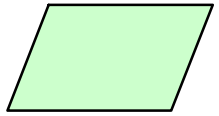
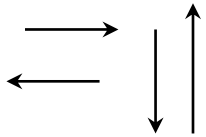
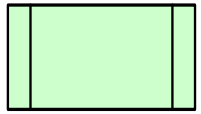
1. Start
2. Prepare a Breakfast
2.1. Prepare a tuna sandwich
2.1.1 Take 2 slices of bread
2.1.2 Prepare tuna paste
2.2. Prepare some chips
2.2.1 Cut potatoes into slices
2.2.2 Fry the potatoes
2.3. Make a cup of coffee
2.3.1 Boil water
2.3.2 Add water with sugar and coffee
3. End

What is a Flow Chart?

- Flowchart is another technique used in designing and representing algorithms.

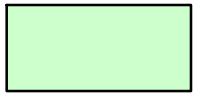
- It is an alternative to pseudocode; whereas a pseudocode description is **verbal** while a flowchart is **graphical** in nature.

- **Defintion:** a graph consisting of geometrical shapes that are connected by flow lines

Symbol	Representation		Symbol	Representation
	Start/Stop			Decision
	Process			Connector
	Input/Output			Flow Direction
	Sub-module / Function (Barred Rectangle)			

This symbol shows where a program start and stop (program or module).

- To identify the beginning and end of a whole program.
- The beginning terminal is labeled with start and the ending terminal is labeled with end or stop.
- When the flowchart is for a module, the beginning terminal is labeled with the module name and the ending terminal is labeled with return or exit.

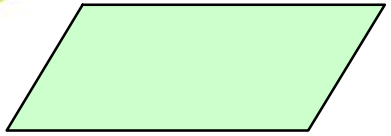


Used to show tasks such as calculations, assignment statements, incrementing, etc.

Processes are labeled with the statement the task to be performed, for example:

totalSales = subTotal + pstAmount + gstAmount
Area = Length * Width
firstName = "Sally"

```
Area = Length * Width
```

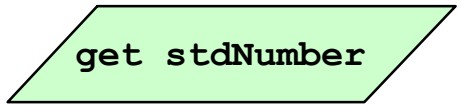


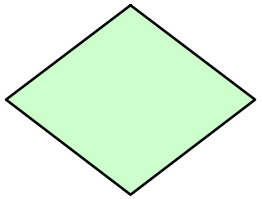
Used to show an operation that brings data into a program, or an operation that sends data out of the program.

- For example, getting values from the user or printing something on the screen.
- Input/output symbols are labeled with the statement that receives the input or generates the output, which could also include opening files and devices.

Examples

get stdNumber

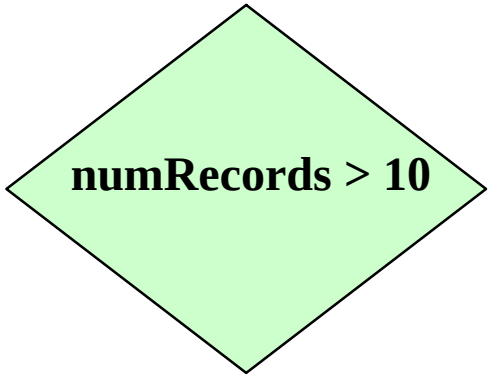




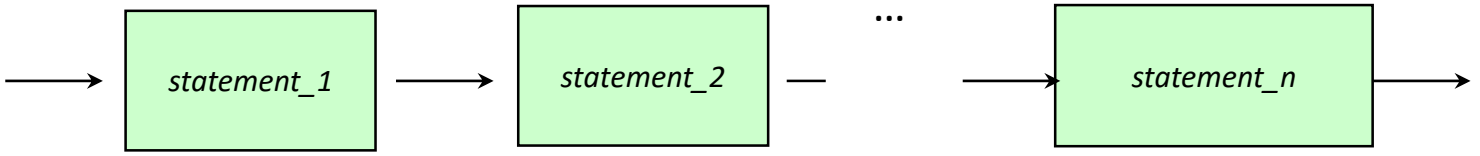
Used to identify a point in the program where a condition is evaluated. Conditions are used in if statements and looping.

- Decisions are labeled with the condition that they are testing, for example:

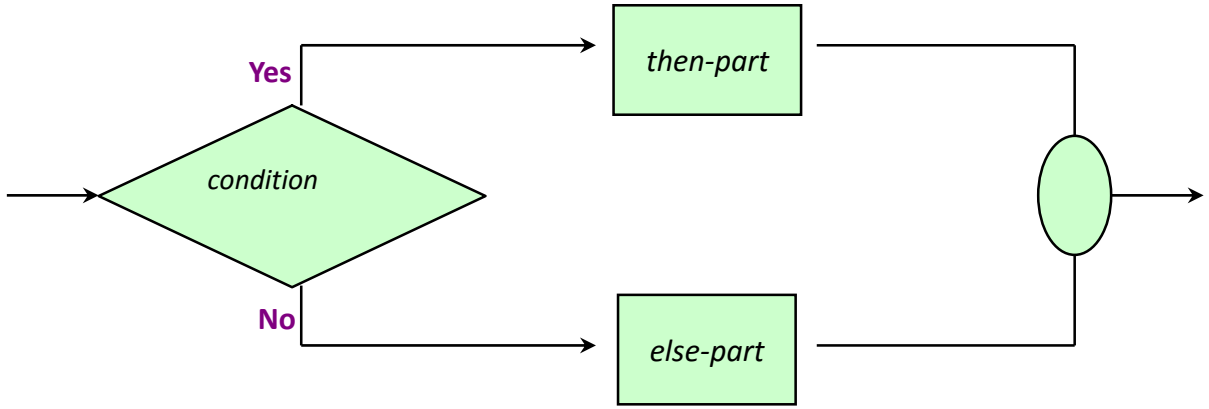
numRecords > 10
foundMatch = true

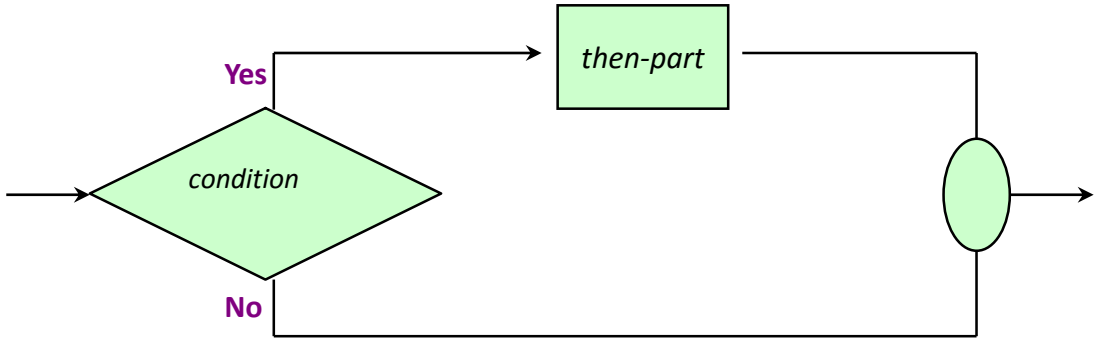


Sequence structure

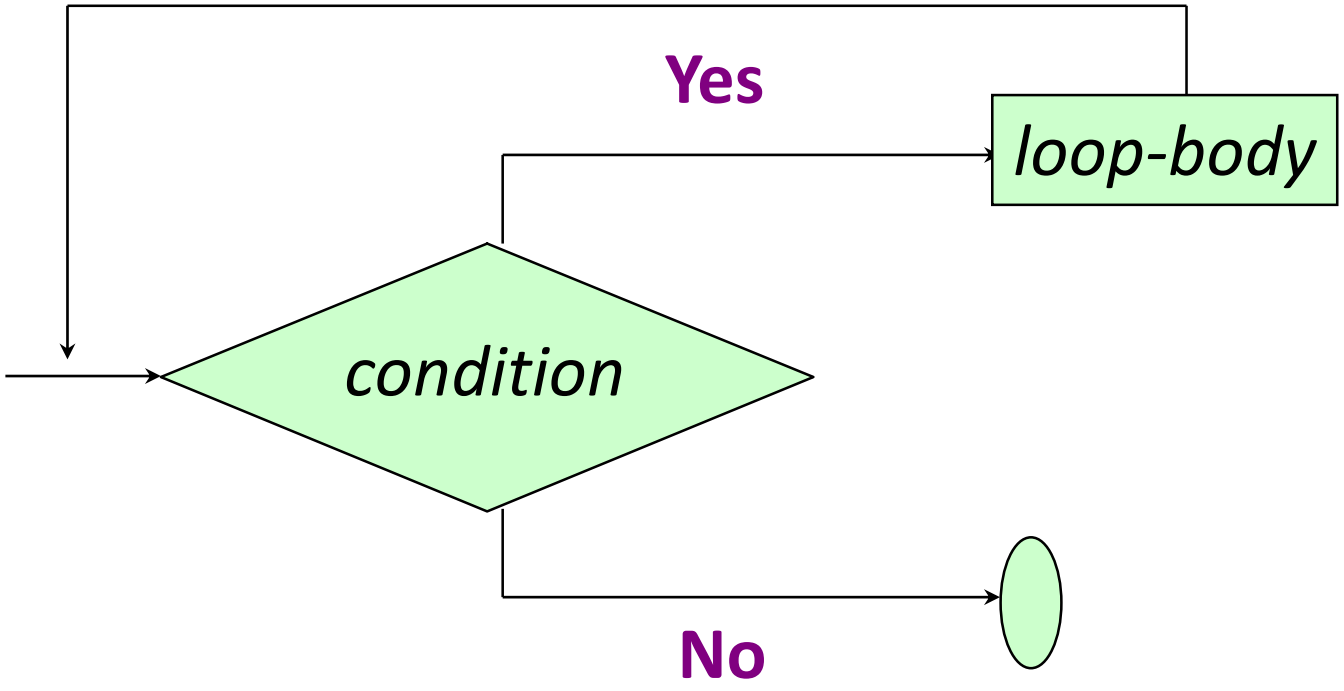


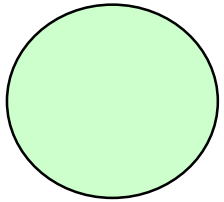
Selection structure





Repetition structure

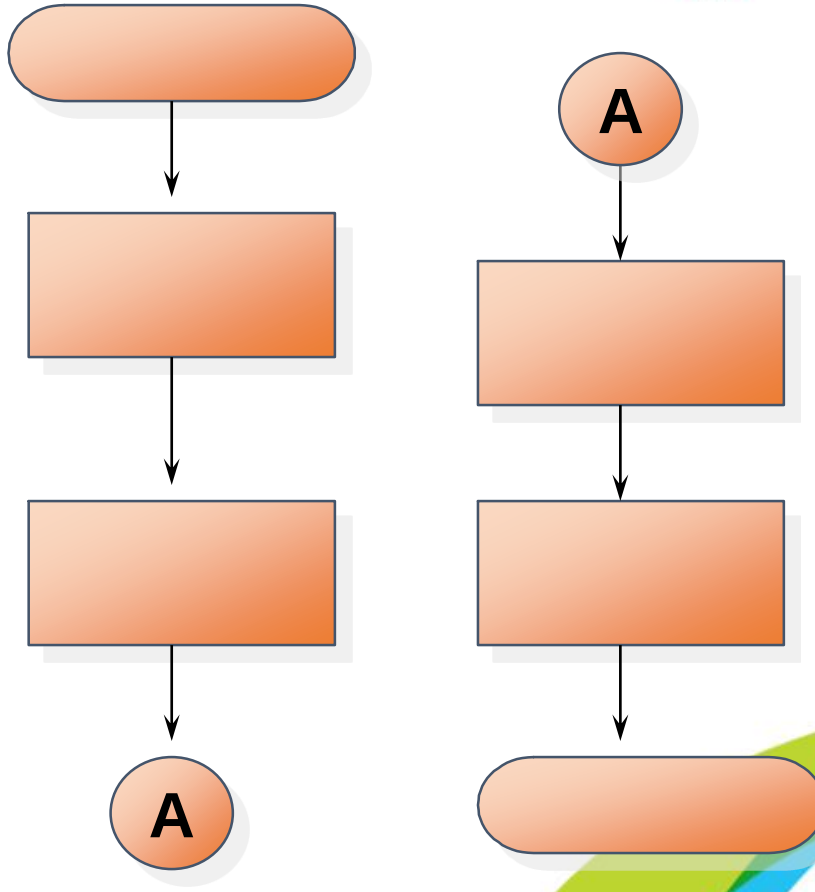


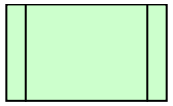


Used to connect two segments of flowchart that appear on the same page.

This is done when your flowchart runs to the bottom of the page and we are out of room: end it with an on-page connector and then place another on-page connector in a free spot on the page, and continue the flowchart from that connector.

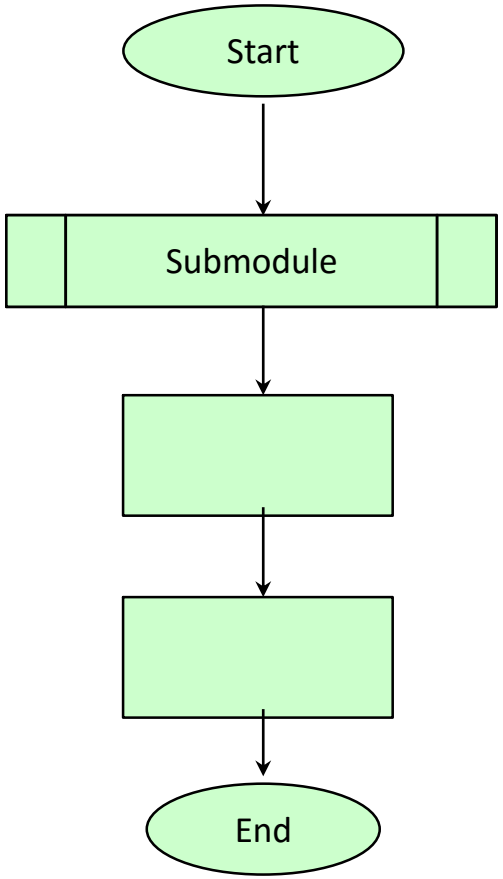
- On-page connectors are labeled with upper-case letters.
- The connector at the end of a segment of flowchart will match the connector that identifies the rest of the flowchart.
- For example, when we run out of room, end the flowchart with a connector labeled "A".
- The flowchart continues at the matching connector also labeled "A".

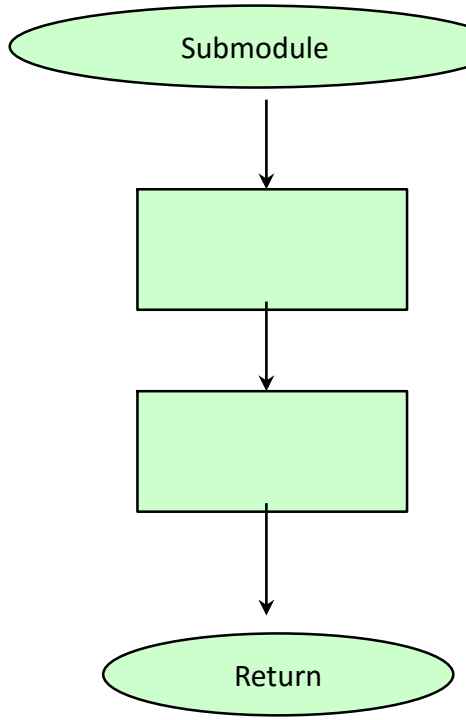




Used to specify a function/module call or a group of related statements.

Example





The benefits of flowcharts

- **Communication**
  - Flowcharts are better way of communicating the logic of a system to all concerned.
- **Effective analysis**
  - With the help of flowchart, problem can be analyzed in more effective way.
- **Proper documentation**
  - Program flowcharts serve as a good program documentation, which is needed for various purposes.
- **Efficient Coding**
  - The flowcharts act as a guide or blueprint during the systems analysis and program development phase.
- **Proper Debugging**
  - The flowchart helps in debugging process.
- **Efficient Program Maintenance**
  - The maintenance of operating program becomes easy with the help of flowchart. It helps the programmer to put efforts more efficiently on that part

Something to ponder …

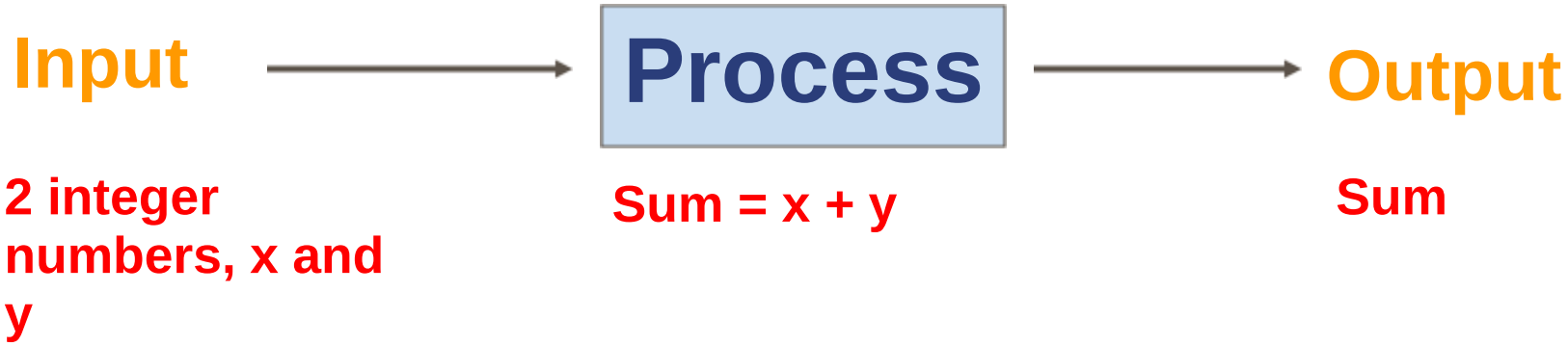
What are the problem solving process?

Problem Solving Process

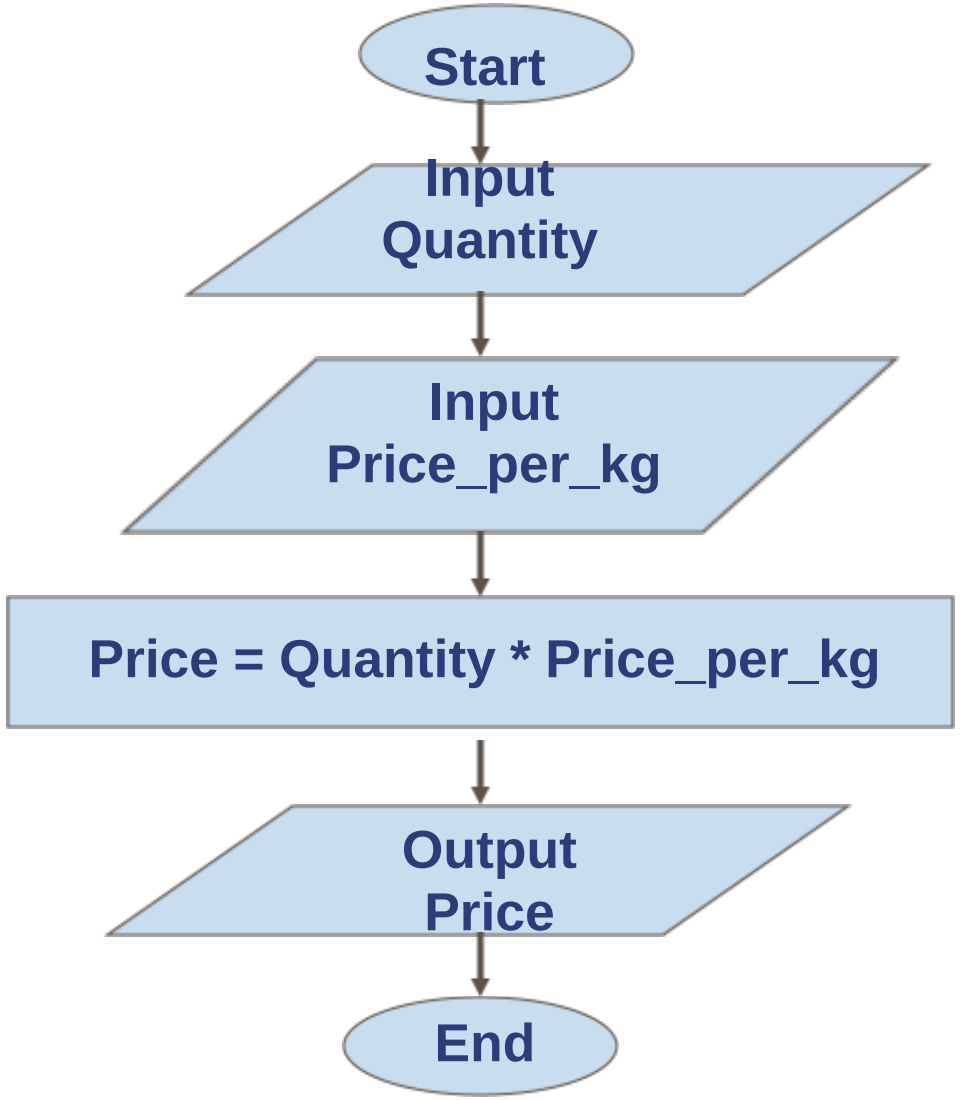
Explanation about the terms

Example 1

Sum of 2 integer numbers



Flowchart: Calculate Price of Apples



Thank You☺

UNIMASMOOC
UNIVERSITI MALAYSIA SARAWAK