# *Developing an Algorithm*

## The Structure Theorem:

The Structured Theorem forms the basic framework for structured programming; it states that it is possible to write any computer program by using only three basic control structures: sequence, selection and repetition

The three basic control structures:

**Sequence:**

The sequence control structure is the straightforward execution of one processing step after another. In an algorithm, we represent this construct as a sequence of pseudocode statements:

**Statement a**
**Statement b**
**Statement c**

Each of these instructions is executed in the order in which it appears. These instructions are usually one of the first four basic operations: to receive information, put out information, perform arithmetic, and assign values.

**Selection:**

The selection control structure is the representation of a condition and the choice between two actions. The choice depending on whether the condition is true or false. This construct represents the decision-making abilities of the computer and represents the fifth operation, namely to compare two variables and selected one of

two alternate actions. In pseudocode, selection is represented by the keywords: IF, THEN, ELSE and ENDIF

**IF condition is true THEN**
**Statement(s) in true case**
**ELSE**
**Statement(s) in false case**
**ENDIF**

If the condition is true then the statement or statements in the true case will be executed, and the statements in the false case will be skipped. Otherwise, (the ELSE statement) the statements in the true case will be skipped and the statements in the false case will be executed.

In either case, control then passes to the next processing step after the delimiter ENDIF.

Example:

IF student is part_time THEN
Add 1 to part_time_count
ELSE
Add 1 to full_time_count
ENDIF

Display part_time_count
Display full_time_count

A variation of the selection control structure is the null ELSE structure, which is used when a task is performed only if a particular condition is true.

The null ELSE construct is written in pseudocode as:

**IF condition is true THEN**
**Statement(s) in true case**
**ENDIF**

Example:

IF today_temperature greater than 95
Display " Another hot day"
ENDIF

Display “Have a nice day”

Note that there is no keyword **ELSE**. This construct tests the condition in the IF clause and if it is found to be true, performs the statement or statements listed in the THEN clause. However, if the initial condition is false, no action will be taken and processing will proceed to the next statement after the ENDIF

**Repetition:**
The repetition control structure can be defined as the presentation of a set of instructions to be performed repeatedly, as long as a condition is true. The basic idea of repetitive code is that a block of statements is executed again and again, until a terminating condition occurs. It is written in pseudocode as:

**WHILE condition is true, DO**

**Statement block**

**ENDWHILE**

WHILE loop is a leading decision loop; that is the condition is tested before any statements are executed.
If the condition in the WHILE loop is found to be true, the block of statements following the condition is executed once. The delimiter ENDWHILE then triggers a return of control to the retesting of the condition. If the condition is still true, the statement block is repeated, and so the repetition process continues until the condition is no longer true.

It is imperative that at least one statement within the statement block can alter the condition and eventually renders it false. Otherwise, the logic may result in an endless loop.

Set student_total to 0

WHILE student_total < 50

Read *Student Record*
Print *Student name, address*
*Add* 1 *to student _ total*

} Statement Block

ENDWHILE

**Points to remember:**

a. The variable student_total is initialized before the WHILE condition is executed.
b. As long as the student_total is less than 50 (the WHILE condition is true), the statement block will be executed.
c. Each time the statement block is executed, one instruction within the block will cause the variable student_total to be incremented.
d. After 50 iteration, student_total will equal 50, which causes the WHILE condition to become false and the repetition to cease.
e. Initialization and subsequent incrementing of the variable tested in the condition is an essential feature of the WHILE construct.

## DEFINING THE PROBLEM:

The first step and one of the most important is defining the problem. To help in this initial analysis, the problem is divided into three separate components:

1. Input: a list of the source data provided to the problem.
2. Output: A list of the output required
3. Processing: a list of actions needed to produce the required outputs.

When reading the problem statement, you can identify the inputs as the data that is either given or read by the program, and outputs are usually the expected result(s) after processing.
Usually, inputs and outputs are expressed as nouns or adjectives.
The actions needed for processing are usually specified in the problem statement as verbs.

When solving a problem:
-Underline all the nouns and adjectives in the problem statement.
-Look at the underlined nouns and decide which are inputs and which are outputs.

Example:
A program is required to read three numbers, add them together and print their total

Our example would look like:

A program is required to read three numbers, add them together and print their total

We use a simple diagram called a ***defining diagram*** to put the information:

| Input | Processing | Output |
|---|---|---|
| | | |

Second, underline (in a different color) the verbs and adverbs used in the specification. This will establish the actions required. The sentence should look like this:

A program is required to read three numbers, add them together and print their total

| Input | Processing | Output |
|---|---|---|
| | | |

Now that all the nouns and verbs in the specification have been considered and the defining g diagram is complete , the problem has been properly defined. That is we now understand the input to the problem, the output to be produced and the processing steps required to convert the input to the output.

**Selecting Meaningful names:**

In defining the problem, it is a good idea to introduce unique names, which will be used to represent the variables in the problem and describe the processing steps. All names should be meaningful. A name given to a variable is simply a method of identifying a particular storage location in the computer. The uniqueness of the name will differentiate it from other locations. The name should be transparent enough to adequately describe the variable.
When it comes to writing down the processing components of the diagram, we use words that describe the work to be done or action words. These actions describe single specific actions:

Read three numbers
Add numbers together
Print total number
Example 2:

Find average temperature:

Write a program that prompts the user for a maximum and a minimum temperature readings on a particular day and calculate and display to the screen the simple average temperature calculated by (maximum temperature + minimum temperature) / 2

First, establish the input and output components:

Write a program that prompts the user for a maximum and a minimum temperature readings on a particular day, get those readings, and calculate and display to the screen the simple average temperature calculated by (maximum temperature + minimum temperature) / 2

| Input | Processing | Output |
|---|---|---|
| | | |

Now, establish, the processing steps by underlying the verbs:

Write a program that prompts the user for a maximum and a minimum temperature readings on a particular day, get those readings, and calculate and display to the screen the simple average temperature calculated by (maximum temperature + minimum temperature) / 2

The processing verbs are : **prompt, get, calculate,** and **display**. By finding the associated objects to these verbs, we specify the defining diagram:

| Input | Processing | Output |
|---|---|---|
| Max_temperature<br>Min_temp | Prompt for temperature<br>Get max, min temperatures<br>Calculate average temperature<br>Display average temperature | Average_temperature |

At this stage, we are only concerned with the fact that the simple average temperature must be calculated and not how the calculation will be performed.

For the following exercises, define the problem by constructing a defining diagram:
1. We require an algorithm that prompts the user for two integers. The program receives the integers and display to the screen their sum, difference, product and quotient.
2. Construct an algorithm that will prompt the user to input three characters, receive those three characters and displays a welcoming message to the screen such as:
"Hello xxx! We hope you have a nice day"

## Designing a solution Algorithm:

Designing a solution algorithm is one of the most challenging tasks in the life cycle of a program. Once the problem has been properly defined, we start with a rough sketch of the steps required to solve the problem.
- Look at the defining diagram
-Using the three structures defined in the theorem (Sequence, Selection and Repetition), attempt to establish how the processing will take place.

-This process is a trial and error process, and you will be adding, deleting or altering an instructions or even discarding the solution and starting gain, until you get a working algorithm.

**IF THE ALGORITHM IS NOT CORRECT, THE PROGRAM NEVER WILL BE.**

## Example 1:

| Input | Processing | total |
|---|---|---|
| number_1<br>number_2<br>number_3 | Read three numbers<br>Add numbers together<br>Print total number | total |

This diagram shows what is required, and a simple calculation will establish how. The solution algorithm looks as follows:

```
Add_three_numbers
        Read number_1, number_2, number_3
        total = number_1 + number_2 + number_3
        Print total
END
```

**Points to remember:**

- A name has been given to the algorithm; Add_three_numbers
- An END statement at the end of the algorithm indicates that the algorithm is complete.
- Each processing step in the defining diagram relates directly to one or more statements in the algorithm.

## Example 2:

A program is required to prompt the user for the maximum and minimum temperature readings on a particular day, accept those readings as integers, and calculate and display to the screen the simple average temperature, calculated by (maximum temperature + minimum temperature) / 2

**Defining diagram:**

| Input | Processing | Output |
|---|---|---|
| max_temp<br><br>min_temp | Prompt for temperatures<br><br>Get max, min temperature<br><br>Calculate average temperature<br><br>Display average temperature | <br>avg_temp |

## Solution algorithm

```
Find_average_temperature
        Prompt user for max_temp, min_temp
```

```
        Get max_temp, min_temp
        avg_temp = (max_temp + min_temp) / 2
        Output avg_temp to the screen
END
```