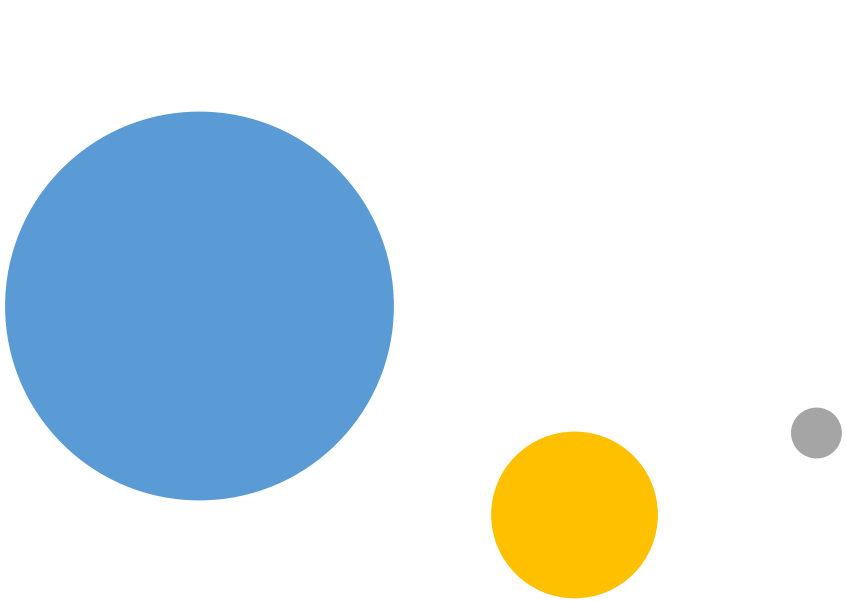


Structure Theorem

Hoan Ng

Traditional Modelling Methods

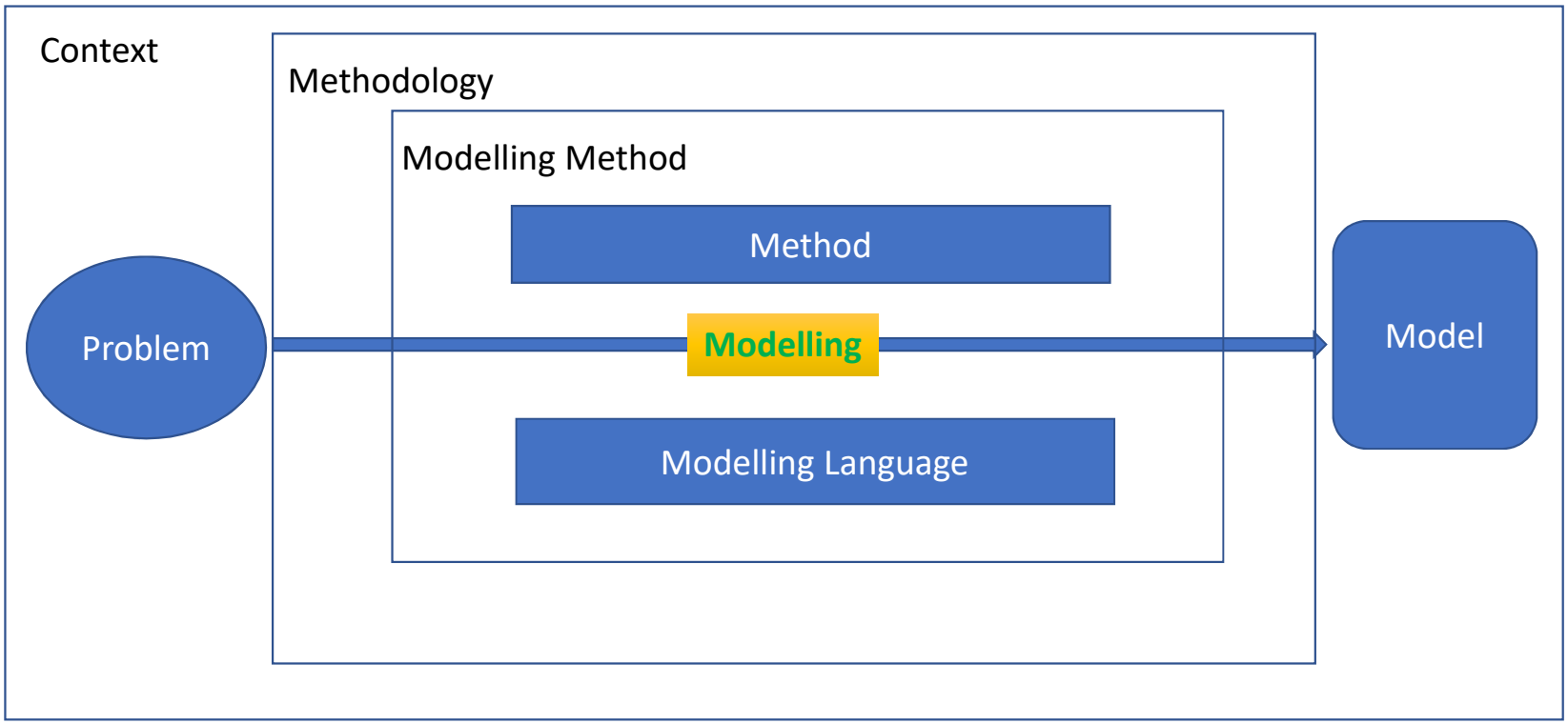
Agenda

Structure Theorem

Basic Rule

Example

Modelling Methods



Structure Theorem

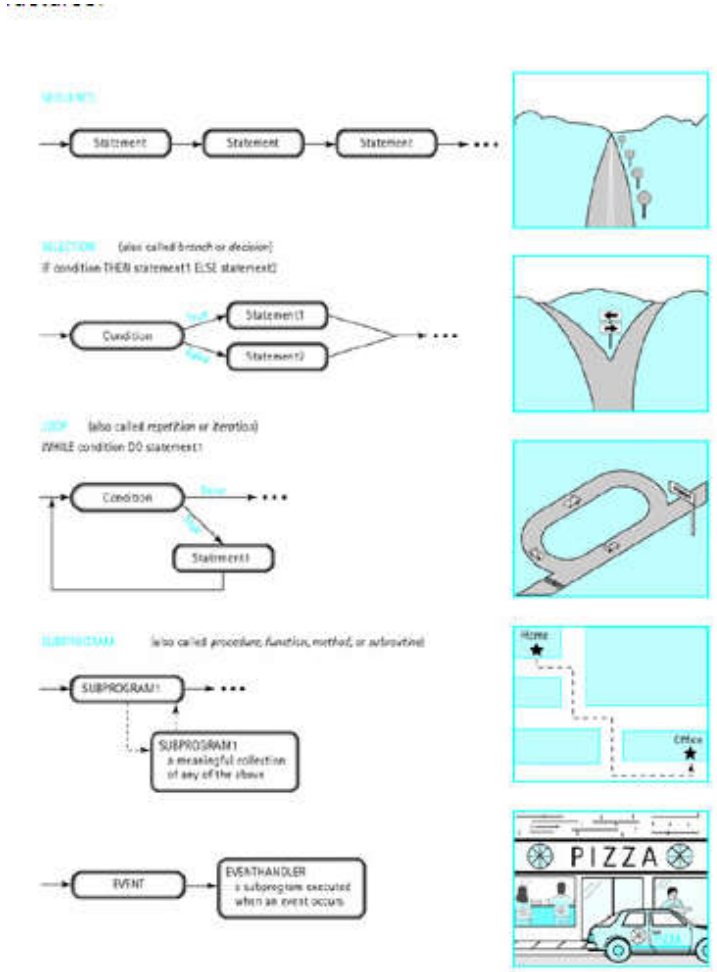


Figure 1.9: Basic control structures of programming languages

Basic Rule

1) Sequence

- Get up from bed
- Dress up
- Go to the shower
- Go to work

2) Selection

- If car is broken, go to work by bus. Otherwise go to work by car

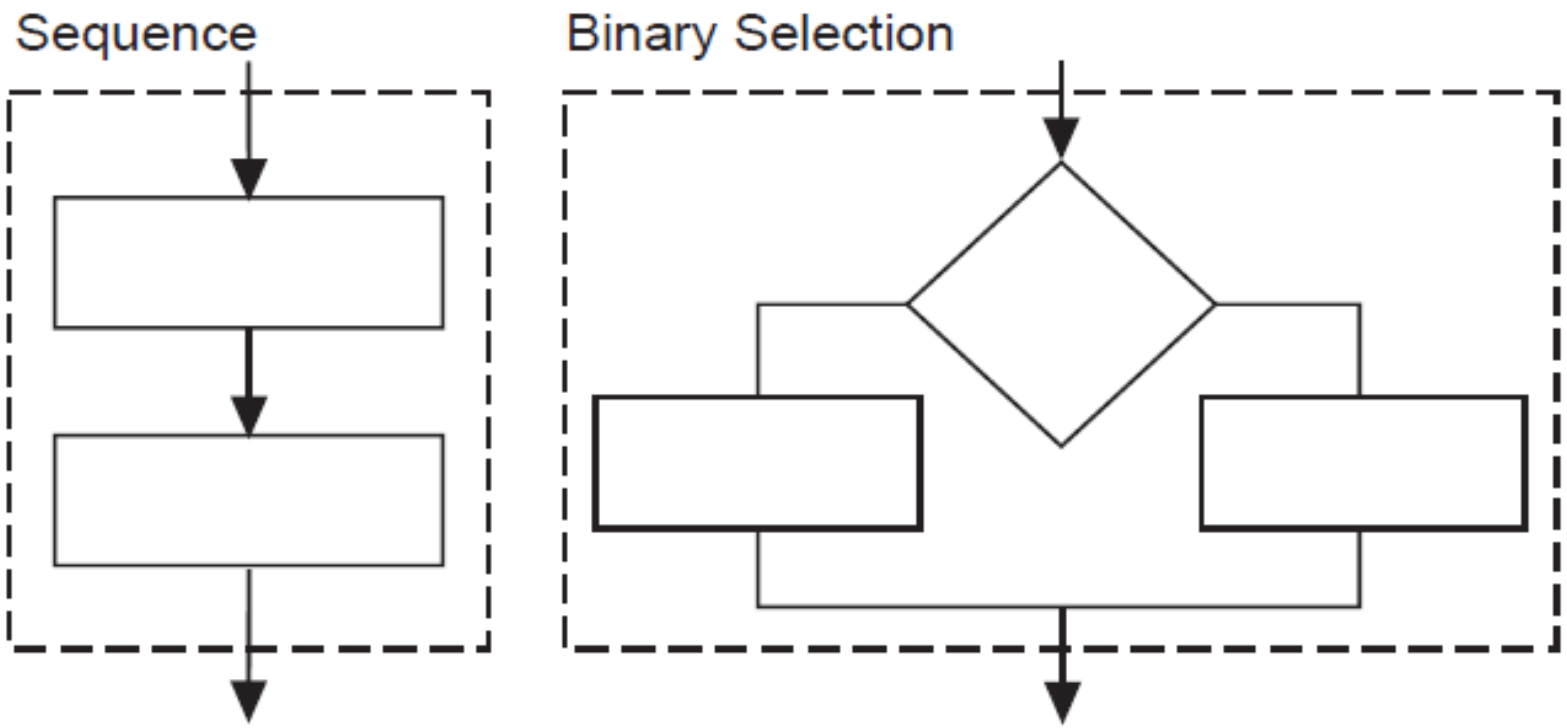
3) Repetition

- Drink until the bottle is empty

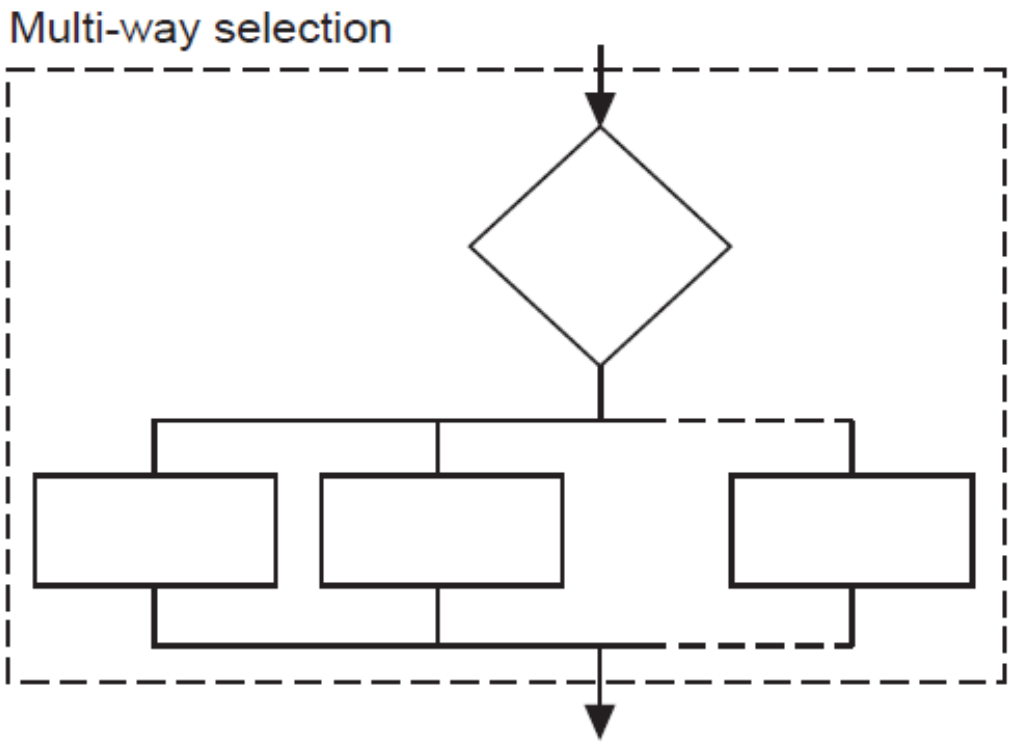
Benefits

- It is considered good practice for a single flowchart
  - never to exceed the bounds of one page.
- If a flowchart does not fit on one page,
  - this is one instance in which the better solution is to use refinement
  - which results in the creation of subprograms.
- Subprograms on separate pages are more desirable than using a connector to join flowcharts over more than one page.
- A flowchart expressing the solution to an involved problem may have:
  - the main program flowchart on one page
  - with subprograms continuing the problem solution on subsequent pages.
- Regardless of page size, it is also important to start any complex algorithm with:
  - a clear, uncluttered main line.
  - This should reference the required subroutines whose detail is shown in separate flowcharts.

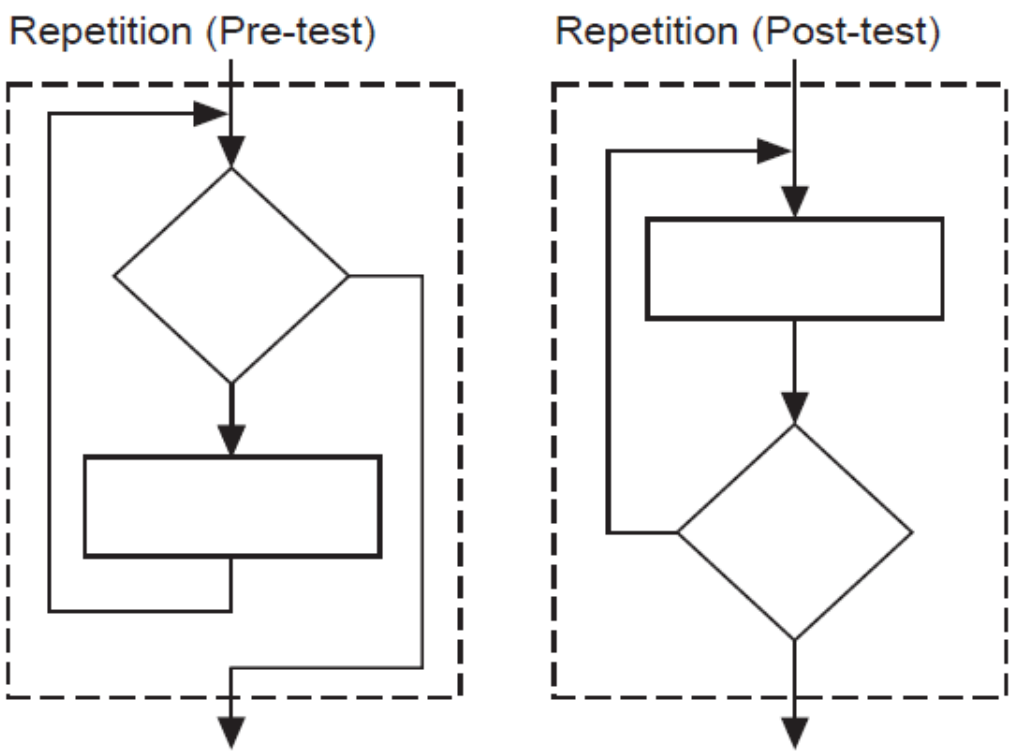
Structure



Structure



Structure



The Structure Theorem

- Each of the five acceptable structures can be built from the basic elements as shown previously
- In all cases note there is:
  - **only one entry point** to the structure
  - and **one exit point** as indicated by the dashed boxes.
-

- Since each structure can be thought of as a process
  - (as shown by the dashed boxes containing the structure),

- more complex algorithms can be constructed by
  - **replacing** any single process **by**
  - one or **other** of the **structures**.

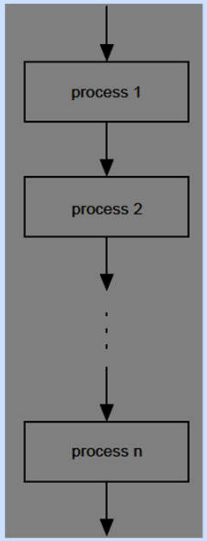
The Structure Theorem - Sequence

In a computer program or an algorithm,
sequence involves simple steps which are
to be **executed one after the other**.
The steps are executed in the same order in which they are written.

In **pseudocode**,
sequence is expressed as:

process 1
process 2
...
...
process n

In a **flowchart**,
sequence is expressed as:
(The arrowheads are optional if the flow is top-to-bottom.)



The Structure Theorem – Sequence Example

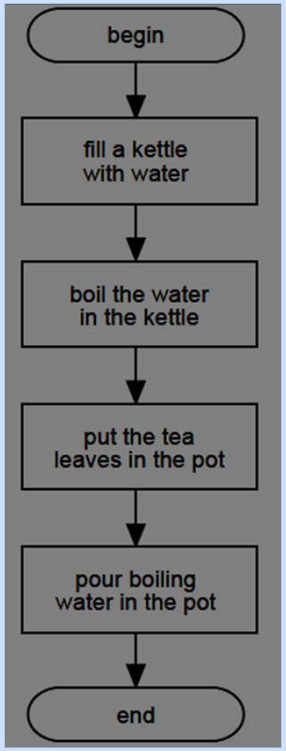
An Example Using Sequence

Problem: Write a set of instructions that describe how to make a pot of tea.

Pseudocode

```
BEGIN
      fill a kettle with water
      boil the water in the kettle
      put the tea leaves in the pot
      pour boiling water in the pot
END
```

Flowchart



The Structure Theorem - Selection

Selection is used in a computer program or algorithm
to **determine which** particular step or set of **steps is to be executed**.

A selection statement can be used to choose a specific path dependent on a condition.
There are two types of selection:

1. **binary** (*two-way branching*) selection and
2. **multi-way** (*many way branching*) selection.

Following is a description of each.

Binary Selection

As the name implies, binary selection allows the **choice between two possible paths.**

If the condition is met then *one path is taken*,
otherwise the *second possible path is followed*.

- In the examples that follow, the first case described requires a **process** to be completed **only if** the condition is **true**.
- The process is **ignored if** the condition is **false**.

In other words there is **only one path** that **requires processing** to be done, so the processing free path is left out rather than included saying 'do nothing'.

Multi-way Selection

Multi-way selection allows for any **number of possible choices,** or cases.

The path taken is determined by the **selection of the choice** which is true.

Multi-way selection is often referred to as a **case structure**.

The Structure Theorem – Binary Selection 1/3

Selection is used in a computer program or algorithm
to **determine which** particular step or set of **step**s **is to be executed**.

Binary Selection

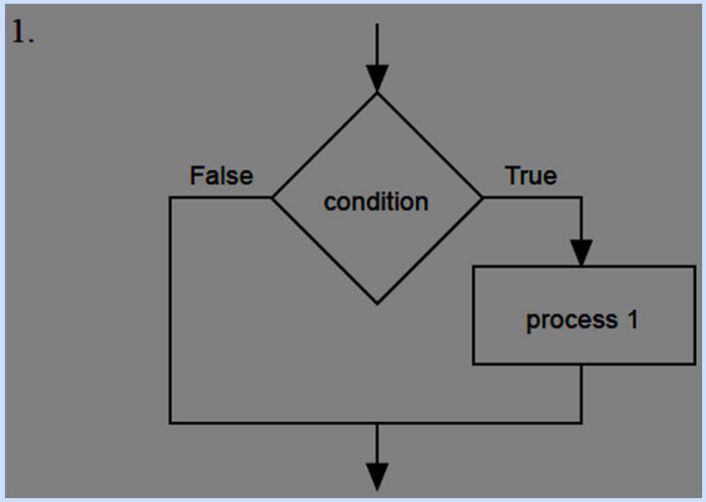
In **pseudocode**, binary selection is expressed in the following ways:

1. IF condition THEN
   process 1
   ENDIF

1. IF condition THEN
   process 1
   ELSE
   process 2
   ENDIF

Binary Selection

In **flowcharts**, binary selection is expressed in the following ways:



The Structure Theorem – Binary Selection 1/3

Selection is used in a computer program or algorithm
to **determine which** particular step or set of **steps is to be executed**.

Binary Selection

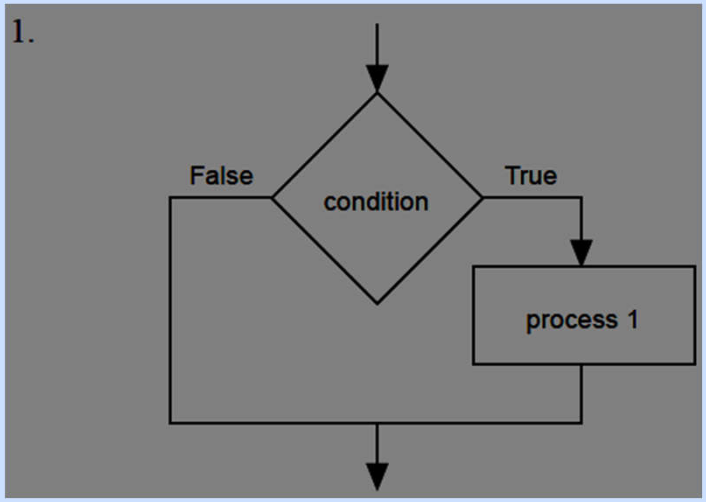
In **pseudocode**, binary selection is expressed in the following ways:

- IF condition THEN
process 1
ENDIF

.IF condition THEN
process 1
ELSE
process 2
ENDIF

Binary Selection

In **flowcharts**, binary selection is expressed in the following ways:



The Structure Theorem – Binary Selection 2/3

Selection is used in a computer program or algorithm
to **determine which** particular step or set of **step**s **is to be executed**.

Binary Selection

In **pseudocode**, binary selection is expressed in the following ways:

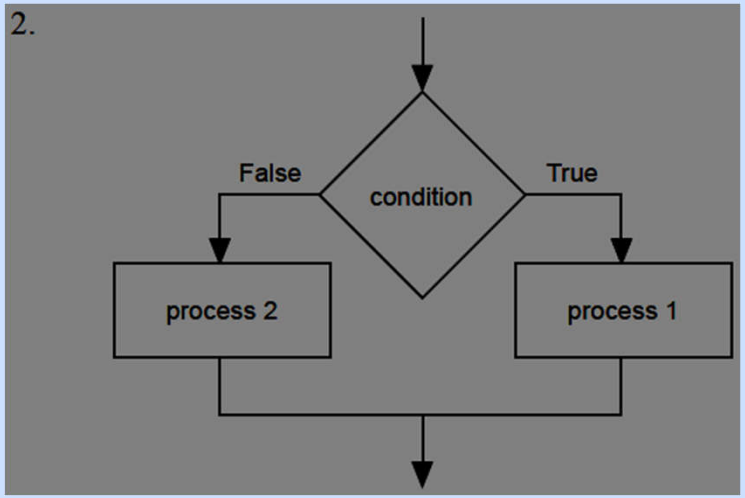
1. IF condition THEN
   process 1
   ENDIF

1. IF condition THEN
   process 1
   ELSE
   process 2
   ENDIF

Binary Selection

In **flowcharts**, binary selection is expressed in the following ways:

2.



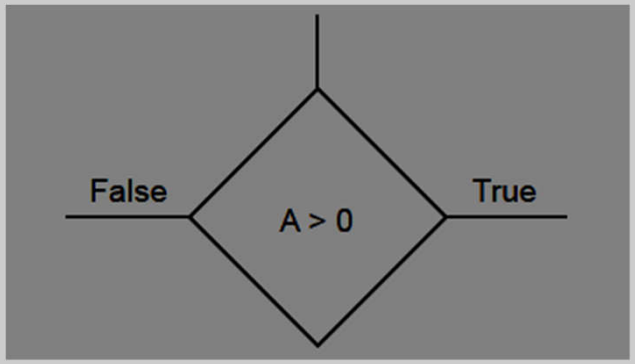
The Structure Theorem – Binary Selection 3/3

Note: In a flowchart it is most important to indicate
1. **which path** is to be followed when the **condition is true**, and
2. **which path** to follow when the **condition is false.**

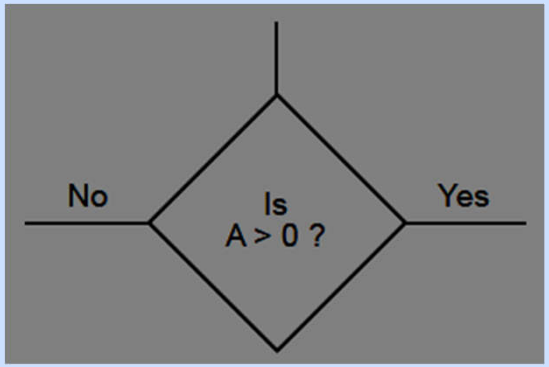
Without these indications the flowchart is open to more than one interpretation.

Note: There are two acceptable ways to represent a decision in all of the structures.
Either method is acceptable. For consistency, the method 1 is used throughout this document.

1. The **condition** is expressed as a **statement** and the two possible outcomes are indicated by
   - True
   - False



2. The **condition** is expressed as a **question** and the two possible outcomes are indicated by
   - Yes
   - No



The Structure Theorem – Binary Selection Examples 1/2

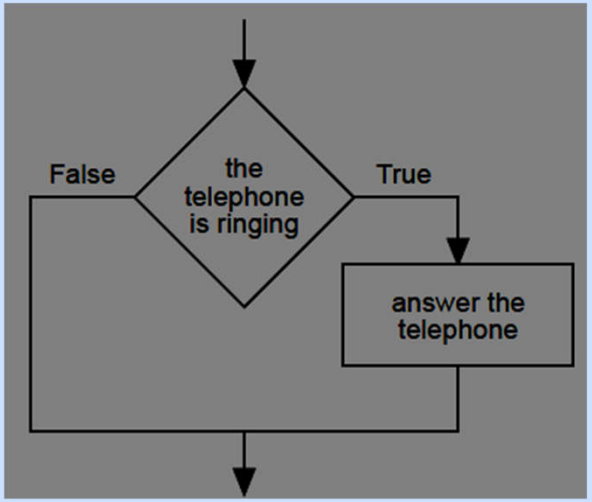
Selection is used in a computer program or algorithm
to **determine which** particular step or set of **step**s **is to be executed**.

Examples Using Binary Selection
Problem 1: Write a set of instructions to describe when to answer the phone.

Binary Selection
Pseudocode

```
IF the telephone is ringing THEN
        answer the telephone
ENDIF
```

Binary Selection
Flowchart



The Structure Theorem – Binary Selection Examples 2/2

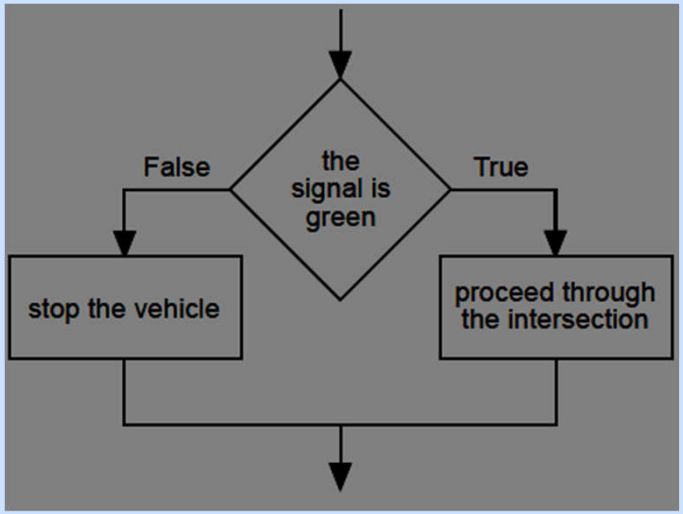
Selection is used in a computer program or algorithm
to **determine which** particular step or set of **steps is to be executed**.

Examples Using Binary Selection
Problem 2: Write a set of instructions to follow when approaching a set of traffic control lights.

Binary Selection
Pseudocode

```
IF the signal is green THEN
        proceed through the intersection
ELSE
        stop the vehicle
ENDIF
```

Binary Selection
Flowchart



The Structure Theorem – Multi-way Selection

Selection is used in a computer program or algorithm
to **determine which** particular step or set of **step**s **is to be executed**.

Multi-way Selection

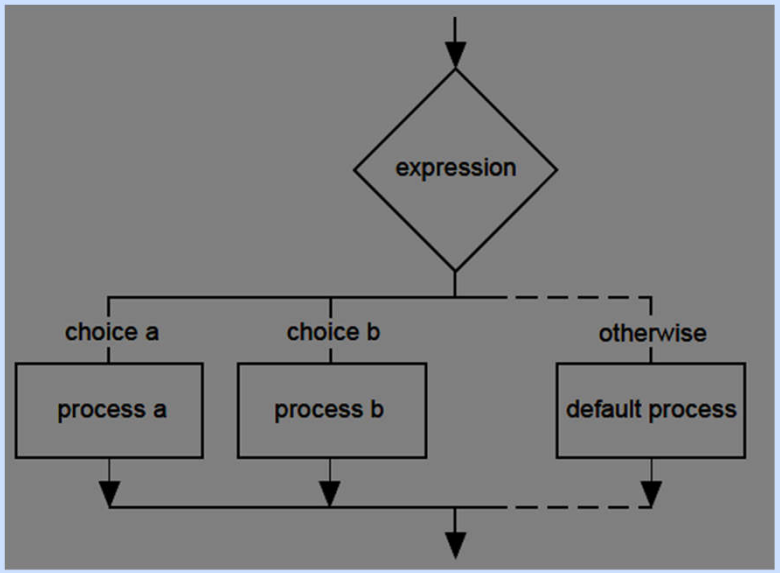
In **pseudocode**, multiple selection is expressed as:

```
CASEWHERE expression evaluates to
        choice a      :      process a
        choice b      :      process b
             .                   .
             .                   .
             .                   .
    OTHERWISE  :      default process
ENDCASE
```

Note: As the flowchart version of the multi-way selection indicates, **only one** process on each pass is executed as a result of the implementation of the multi-way selection.

Multi-way Selection

In **flowcharts**, multi-way selection is expressed as:



The Structure Theorem – Multi-way Selection Examples

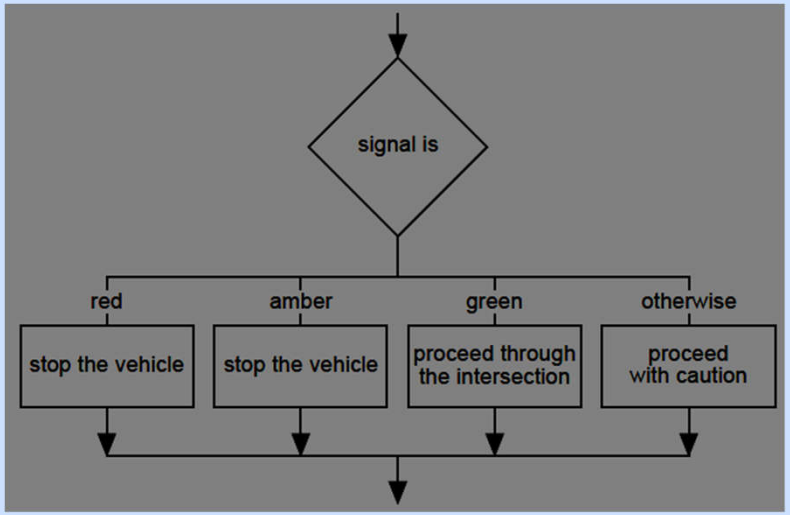
Selection is used in a computer program or algorithm
to **determine which** particular step or set of **step**s **is to be executed**.

Example Using Multi-way Selection
Problem: Write a set of instructions that describes how to:
respond to all possible signals at a set of traffic control lights.

Multi-way Selection
Pseudocode

```
CASEWHERE signal is
        red             : stop the vehicle
        amber           : stop the vehicle
        green           : proceed through the
        intersection
    OTHERWISE  : proceed with caution
ENDCASE
```

Multi-way Selection
Flowchart



The Structure Theorem – Repetition 1/3

Repetition allows for a **portion of** an **algorithm** or computer program
to be done **any number of times**
dependent on some **condition being met**.
An occurrence of repetition is usually known as **a loop**.

An **essential feature** of repetition is that
each loop has a **termination condition**
to **stop** the repetition,
or the obvious outcome is that
the *loop never completes* execution (*an infinite loop*).

The **termination condition** can be checked or **tested**

1. at the **beginning** and is known as a **pre-test** loop *or*
2. at the **end** of the loop and is known as a **post-test** loop.

The Structure Theorem – Repetition 2/3

Repetition allows for a **portion of** an **algorithm** or computer program
to be done **any number of times**
dependent on some **condition being met**.
An occurrence of repetition is usually known as **a loop**.

Repetition: Pre-Test

A pre-tested loop is so named because the **condition** has to be met at the **very beginning of the loop** or the body of the loop is not executed.
This construct is often called a *guarded loop.*
The body of the loop is executed repeatedly while the termination condition is true.

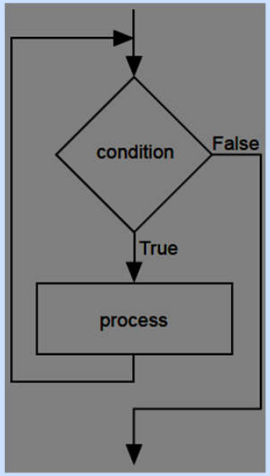
Repetition

In **pseudocode**, pre-test repetition is expressed as:

```
WHILE condition is true
        process(es)
ENDWHILE
```

Repetition

In **flowcharting** pre-test repetition is expressed as:



The Structure Theorem – Repetition 3/3

Repetition allows for a **portion of** an **algorithm** or computer program
to be done **any number of times**
dependent on some **condition being met**.
An occurrence of repetition is usually known as **a loop**.

Repetition: Post-Test

- A post-tested loop **executes the body** of the loop **before testing** the **termination condition**.
- This construct is often referred to as an *unguarded loop.*
- The body of the loop is repeatedly executed until the termination condition is true.

An **important difference** between a pre-test and post-test loop is that the **statements of a post-test loop** are **executed at least once** even if the condition is originally true, whereas the **body of the pre-test loop** *may never be executed* if the *termination condition is originally true*.
A close look at the representations of the two loop types makes this point apparent.

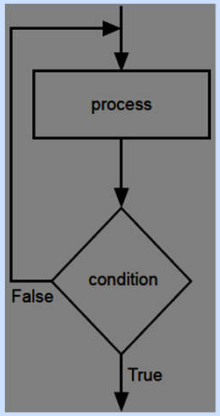
Repetition

In **pseudocode**, post-test repetition is expressed as:

```
REPEAT
        process
UNTIL condition is true
```

Repetition

In a **flowchart** post-test repetition is expressed as:



The Structure Theorem – Repetition Examples 1/2

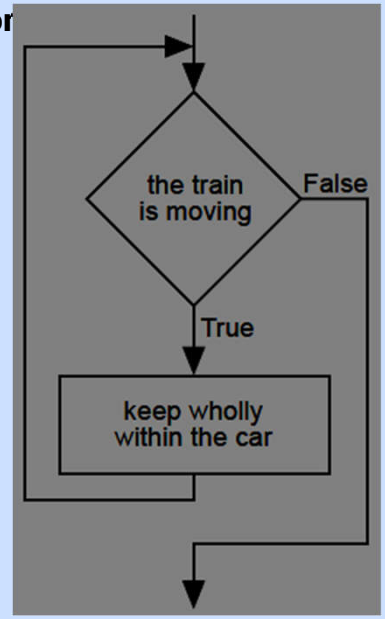
Repetition allows for a **portion of** an **algorithm** or computer program
to be done **any number of times**
dependent on some **condition being met**.
An occurrence of repetition is usually known as **a loop**.

An Example Using Pre-Test Repetition
Problem: Determine a safety procedure for travelling in a carriage on a moving train.

Pre-test Repetition
Pseudocode

```
WHILE the train is moving
        keep wholly within the carriage
ENDWHILE
```

Pre-test Repetition
Flowchart



The Structure Theorem – Repetition Examples 2/2

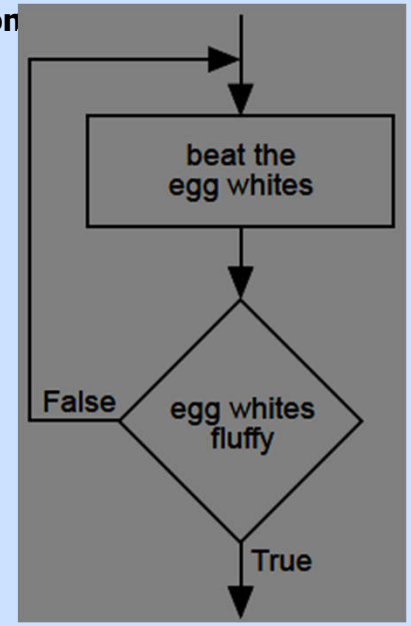
Repetition allows for a **portion of** an **algorithm** or computer program
to be done **any number of times**
dependent on some **condition being met**.
An occurrence of repetition is usually known as **a loop**.

An Example Using Post-Test Repetition
Problem: Determine a procedure to beat egg whites until fluffy.

Post-test Repetition
Pseudocode

```
REPEAT
        beat the egg whites
UNTIL fluffy
```

Post-test Repetition
Flowchart



The Structure Theorem – Subprograms 1/3

Subprograms, as the name implies, are **complete part-programs**
that are used from **within the main program** section.

- They allow the process of **refinement** to be used to develop solutions to problems that are easy to follow.
- Sections of the solution are developed and presented in **understandable chunks**, and because of this,
- subprograms are particularly useful when using the **top-down** method of solution development.

When using subprograms it is important that
1. the solution expression indicates **where the main program branches to a subprogram**.
2. It is equally important to indicate exactly **where the subprogram begins**.

In **pseudocode,**
the **statement in the main program** that is expanded in a subprogram is **underlined** to indicate that
further explanation follows.
The expanded subprogram section should be identified by
1. using the keywords BEGIN SUBPROGRAM
- followed by the underlined title used in the main program.
2. The end of the subprogram is marked by the keywords END SUBPROGRAM
- and the underlined title used in the main program.

When using **flowcharts**,
a subprogram is shown by **an additional vertical line** on **each side of the process box**.
This indicates that the subprogram is **expanded elsewhere**.
The **start** and **end** of the subprogram flowchart **uses** the **name** of the **subprogram** in the **termination boxes**.

The Structure Theorem – Subprograms 2/3

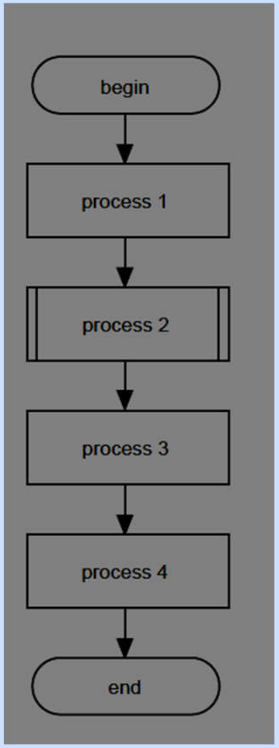
Subprograms, as the name implies, are **complete part-programs** that are used from **within the main program** section.

Subprograms
Pseudocode

```
BEGIN MAINPROGRAM
        process l
        process 2
        process 3
        process 4
END MAINPROGRAM

BEGIN SUBPROGRAM process 2
        do this
        do that
END SUBPROGRAM process 2
```

Subprograms
Flowchart



The Structure Theorem – Subprograms 3/3

Subprograms, as the name implies, are **complete part-programs**
that are used from **within the main program** section.

In many cases a subprogram can be written to **do the same task** at two or **more points in an algorithm**.
- Each time the subprogram is called, *it may operate on different data*.
- To indicate the data to be used one or more **parameters** are used.

The parameters allow the author to **write a general algorithm** using the formal parameters.
When the subprogram is executed, the algorithm carries out its task on the **actual parameters** given at the call.

The parameters to be used by a subprogram are provided as a **list in parentheses** *after the name of the subprogram.*
There is no need to include them at the end of the algorithm.

Example of Using Subprograms with one Parameter in Pseudocode

```
BEGIN MAINPROGRAM
        read (name)
        read (address)
END MAINPROGRAM

BEGIN SUBPROGRAM read (array)
        Set pointer to first position
        Get a character
        WHILE there is still more data AND there is room in the array
                store data in the array at the position given by the pointer
                Increment the pointer
                get data
        ENDWHILE
END SUBPROGRAM read (array)
```

The first time that the subprogram *'read'* is called:
1. the characters are read into the array called *'name'*
2. the second time, the data (characters) are read into the array called *'address'.*