VIET NAM NATIONAL UNIVERSITY HO CHI MINH CITY

Programming for Engineers

Lecture 2A: Flowchart, Pseudo Code, Decision Making

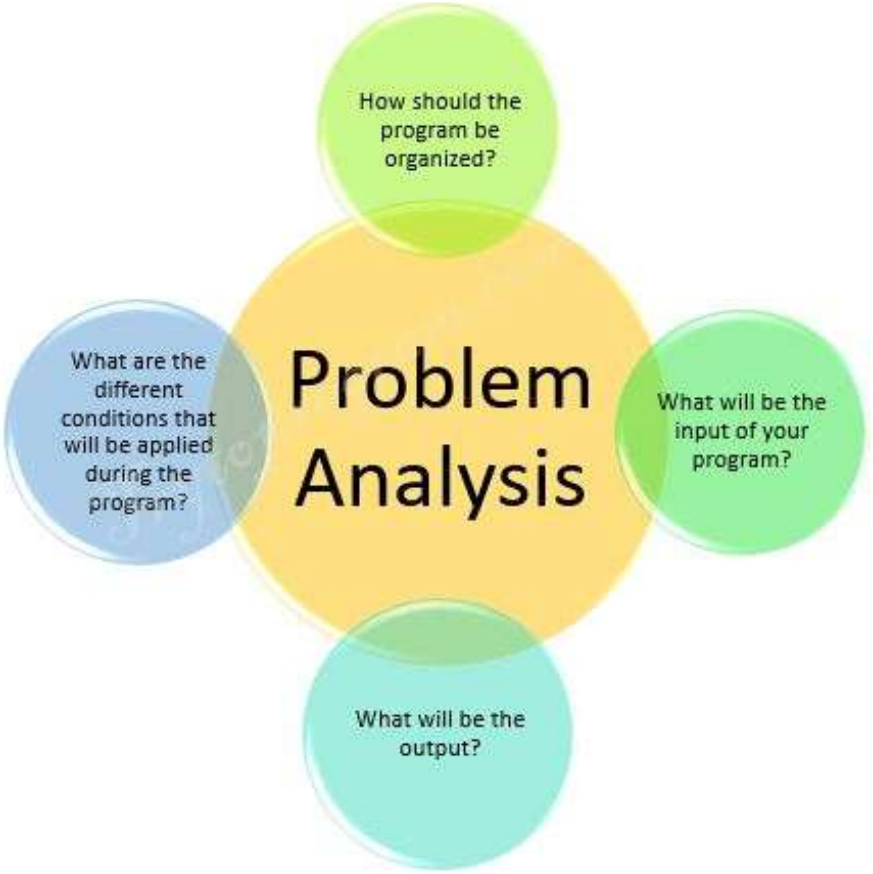
Course ID: EE057IU

Lecture Outline

1. **Flowcharts & Pseudocode**
   - Tools for visualizing and planning code.
2. **Algorithms**
   - Key steps and importance of order in execution.
3. **Why Flowcharts & Pseudocode Matter**
   - Simplify problems and improve communication.
4. **Decision-Making**
   - if, if-else, and switch statements for logical flow in programs.

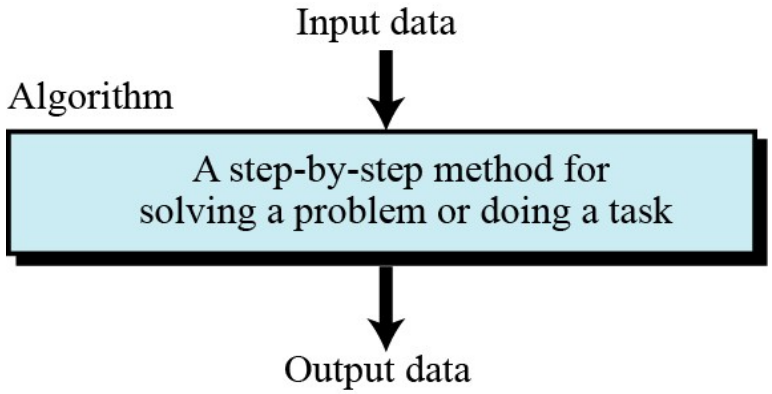
Steps before writing a program

- Understanding the problem
- Carefully planning solution approach
  - Step by step procedure
  - Chapter 3 and 4 cover how to develop structured computer program
  - → **Flowchart** and **Pseudo code**



Algorithms

- The solution to any computer problem which involves executing a series of actions in a specific order
- A procedure for solving a problem in terms of
  - The actions to be executed
  - The orders in which the actions are to be executed



Order matters

- Example “rise-and-shine” algorithm

In-Order	
1. Get out of bed	
2. Take off pajamas	
3. Take the shower	
4. Get dressed	
5. Eat breakfast	
6. Go to school	

Order matters

➢Example “rise-and-shine” algorithm

In-Order	Out-of-Order
1. Get out of bed	1. Get out of bed
2. Take off pajamas	2. Take off pajamas
3. Take the shower	3. Get dressed
4. Get dressed	4. Take the shower
5. Eat breakfast	5. Eat breakfast
6. Go to school	6. Go to school

➢ Specifying the order in which statements should execute in a computer program is called program control.

Order matters

```c
#include <stdio.h>

int main() {
   int a = 2;
   int b = 3;
   int c = 4;
   int result;

   // Correct order: Add first, then multiply
   result = (a + b) * c;

   printf("Result (correct order): %d\n", result);

   return 0;
}
```

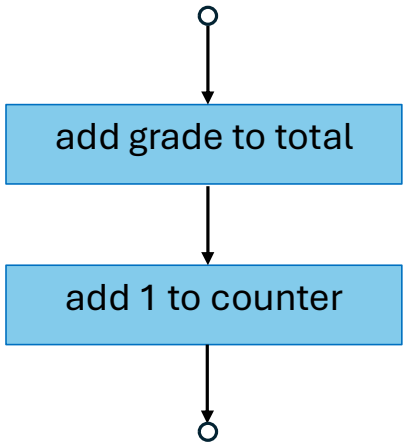
Result (correct order): 20

```c
#include <stdio.h>

int main() {
   int a = 2;
   int b = 3;
   int c = 4;
   int result;

   // Incorrect order: Multiply first, then add
   result = a + b * c;

   printf("Result (incorrect order): %d\n", result);

   return 0;
}
```

Result (incorrect order): 14

Flowchart

- Graphical representation of an algorithm
- Uses certain special-purpose symbols such as rectangles, diamonds, rounded rectangles, and small circles
- Symbols are connected by arrows called flowlines



total = total + grade

counter = counter + 1

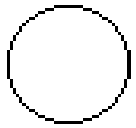
Flowchart - Symbols

Start or end of the program

Computational steps or processing function of a program

Input or output operations

Decision making and branching

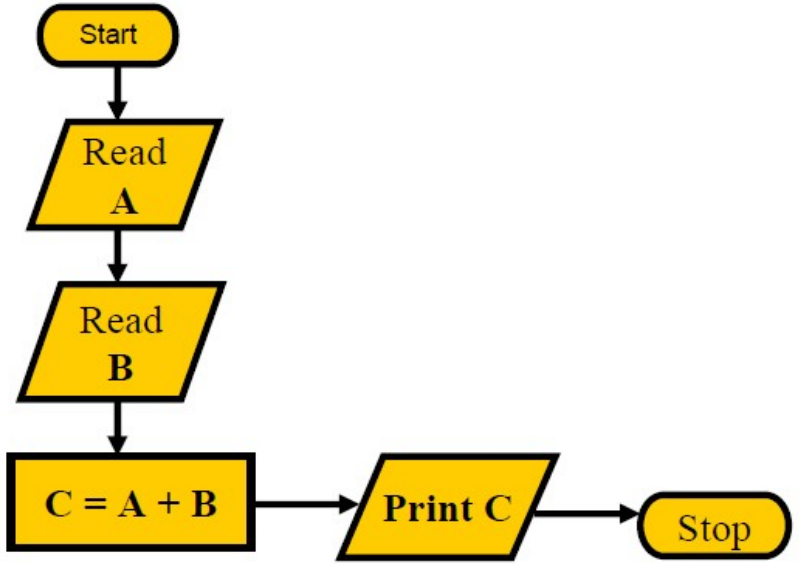


Connector or joining of two parts of program

Flowchart – Example 1

Find the sum of two numbers.

- Variables:
  - A: First Number
  - B: Second Number
  - C: Sum (A+B)
- Algorithm:
  - Step 1 – Start
  - Step 2 – Input A
  - Step 3 – Input B
  - Step 4 – Calculate C = A + B
  - Step 5 – Output C
  - Step 6 – Stop

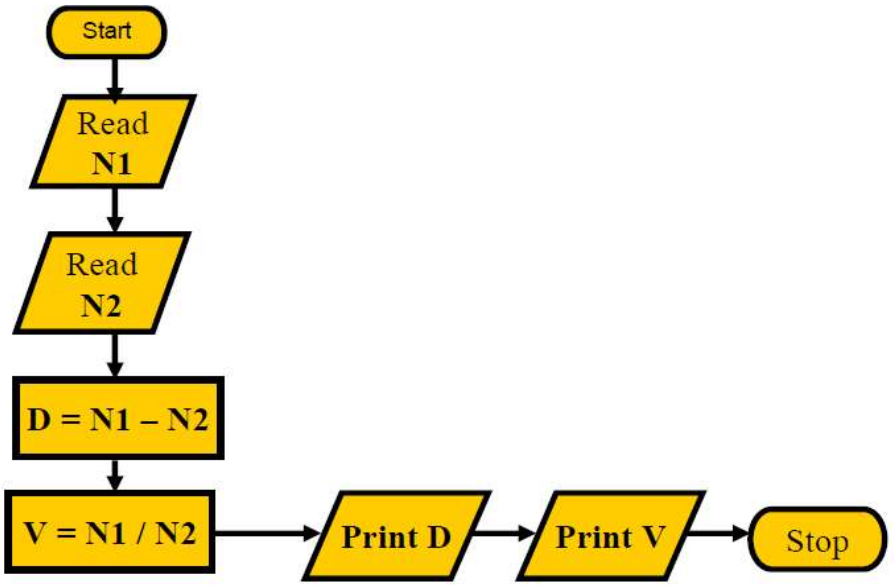


Flowchart – Example 2

Find the difference, the division of two numbers and display the results.

– Variables:
- N1: First number
- N2: Second number
- D : Difference
- V : Division

-- Algorithm:
* Step 1: Start
* Step 2: Input N1
* Step 3: Input N2
* Step 4: D = N1 – N2
* Step 5: V = N1 / N2
* Step 6: Output D
* Step 7: Output V
* Step 8: Stop



Pseudocode

- Artificial, informal, user-friendly, convenient, English-like
- They are NOT executed on computers
- Can be easily converted or translated to ANY programming language
- Consists of *actions* and *decisions*

Example

1. ***Set total to zeros***
2. ***Set grade counter to one***
3.
4. ***While grade counter is less than or equal to ten***
5. ***Input the next grade***
6. ***Add the grade into the total***
7. ***Add one to the grade counter***
8.
9. ***Set the class average to the total divided by ten***
10. ***Print the class average***

Control Structures

➢Sequential execution: statements are executed one after another

→ Previous examples

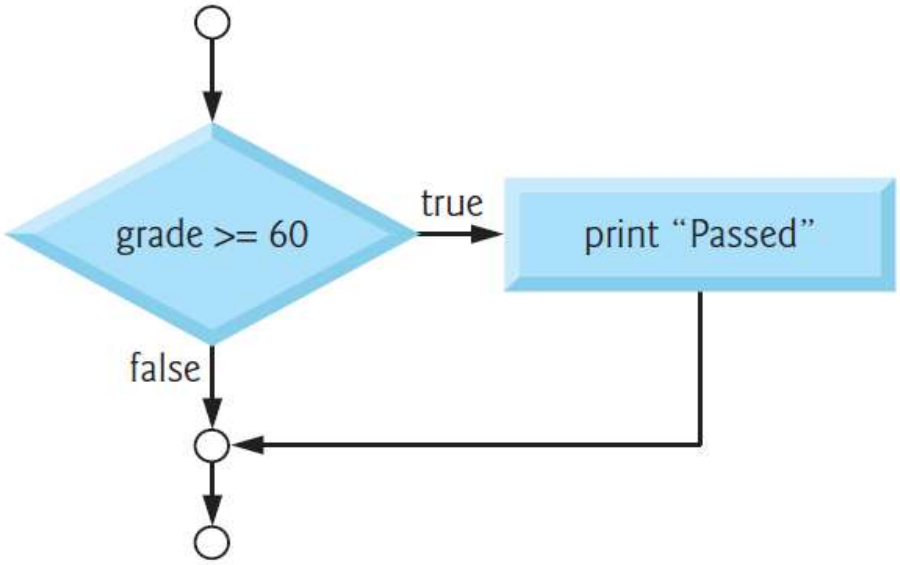
➢Transfer of control: specify next C statement to be executed MAY NOT be the next one in sequence

→ **Decision making**:

- **If** single-selection statement (Chapter 3.5)
- **If … else** double-selection statement (Chapter 3.6)
- Relational & Equality Operators (Chapter 2.6)
- **Switch** multiple statement (Chapter 4.6)

Decision making – If statement

1. Flowchart



- If condition True, statement is executed
- If condition False, statement is not executed

2. Pseudocode

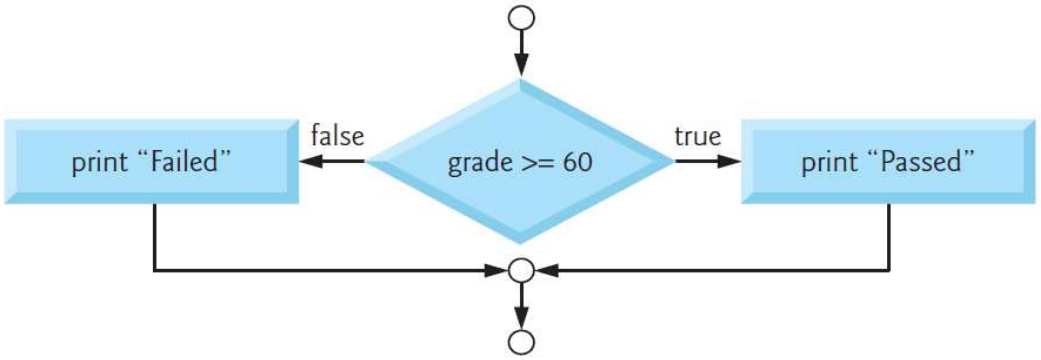
1. *If student's grade is greater than or equal to 60*
2. *Print "Passed"*

3. C statements

```
1. if (grade >= 60)
2. {
3.     printf("Passed\n");
4.  }
```

Decision making – If … else statement

1. Flowchart



- If condition True, one statement is executed
- If condition False, different statement is executed

2. Pseudocode

```
If student's grade is greater than or equal to 60
    Print "Passed"
else
    Print "Failed"
```

3. C statements

```
if (grade >= 60)
{
    printf("Passed\n");
}
else
{
    printf("Failed\n");
}
```

Decision making - Relational & Equality Operators

Algebraic equality or relational operator	C equality or relational operator	Sample C condition	Meaning of C condition
	Relational operators		
>	>	x > y	x is greater than y
<	<	x < y	x is less than y
≥	>=	x >= y	x is greater than or equal to y
≤	<=	x <= y	x is less than or equal to y
	Equality operators		
=	==	x == y	x is equal to y
≠	!=	x != y	x is not equal to y

Decision making - Precedence of Operators

Operators	Associativity
()	left to right
* / %	left to right
+ -	left to right
< <= > >=	left to right
== !=	left to right
&&	left to right
||	left to right
=	left to right

Decision making - Conditional Operator (?)

- **?** Closely related to if … else statement
- **?** Takes three operands
  - First one is *condition*
  - Second one is value/statement when *condition* is TRUE
  - Third one is value/statement when *condition* is FALSE

→ **Example**

printf(grade >= 60 ? “Passed” : “Failed”);

C statements

```
1.  if (grade >= 60)
2.  {
3.      printf("Passed\n");
4.   }
5.   else
6.  {
7.      printf(“Failed\n");
8.  }
```

Decision making – Nested If … else, else if

Nested If ... else statement

```
if ( grade >= 90 )
   puts( "A" );
else
   if ( grade >= 80 )
      puts("B");
   else
      if ( grade >= 70 )
         puts("C");
      else
         if ( grade >= 60 )
            puts( "D" );
         else
            puts( "F" );
```

else if statement

```
if ( grade >= 90 )
   puts( "A" );
else if ( grade >= 80 )
   puts( "B" );
else if ( grade >= 70 )
   puts( "C" );
else if ( grade >= 60 )
   puts( "D" );
else
   puts( "F" );
```

Decision making – Compound statement

- one statement in **If**, **else … if**, **else**'s body, no need braces { and }
- Several statements, need to use braces { and }

```
If (condition1)
    statement1;
else if (condition2)
    statement2;
else if (condition3)
{
    statement3;
    statement4;
}
else
{
    statement5;
    statement6;
}
```

Example:

```
if (grade >= 60)
    puts("Passed.");
else
{
    puts("Failed.");
    puts("You must take this course again.");
}
```

In-class Practice 1

- **Statement**
  - Provide two integers **int1** and **int2**
  - Compare **int1** and **int2**
    - if **int1** > **int2**, print the string Integer 1 is larger than integer 2
    - if **int1** < **int2**, print the string Integer 1 is smaller than integer 2
    - if **int1** = **int2**, print the string Integer 1 is equal to integer 2

→ Requirement:

1. Write a Pseudocode
2. Draw a flowchart
3. Write a C Program

In-class Practice 2

- **Statement**
  - Provide an arbitrary **int**
  - Check whether **int** is:
    - Positive, print “The number is positive”
    - Negative, print “The number is negative”
    - Equal to 0, print “The number is zero”

In-class Practice 3

- **Statement**
  - Provide three integers **int1**, **int2**, **int3**
  - Determine:
    - Largest value of the three integers
    - Calculate the average of the three integers