CSE 06131223 ♦ CSE 06131224

Structured Programming

Lecture 3

Computer Programming Fundamentals (2)

Prepared by

Md. Mijanur Rahman, Prof. Dr.
Dept. of Computer Science and Engineering
Jatiya Kabi Kazi Nazrul Islam University, Bangladesh
www.mijanrahman.com

Contents

Computer Programming Fundamentals

- Introduction
- Software and Programming Language
- Types of Programming Languages
- Commonly Used Programming Languages
- Program Development Life Cycle
- Types of Errors in Programs
- Algorithm and Flowchart
- Control Structures
- Pseudo Code
- Programming Paradigms

Algorithm

- In computer programming terms, **an algorithm is a set of well-defined instructions to solve a particular problem**. It is a step-by-step procedure for solving a task or a problem. It takes a set of input and produces a desired output.
- This method of solution should be designed in a programming manner. Before starting to write the code of a computer program, a programmer has to decide and design steps that is required to solve the problem; **this is called designing the algorithm of the proposed program.**
- Thus, **an *algorithm* is an ordered sequence of finite, well defined, unambiguous instructions** for completing a task. It is an English-like representation of the logic which is used to solve the problem.

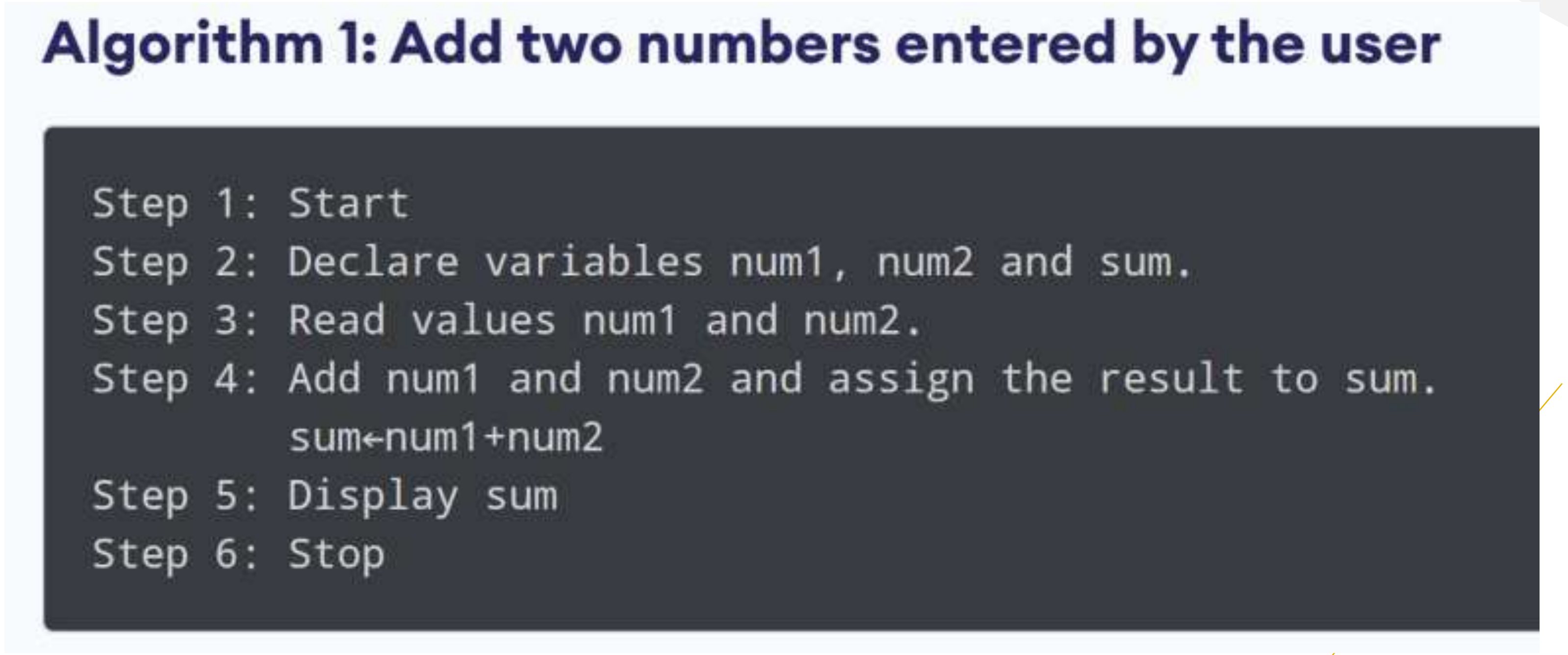
Algorithm

- For accomplishing a particular task, different algorithms can be written. The different algorithms differ in their requirements of time and space. **The programmer selects the best-suited algorithm for the given task to be solved.**
- **Algorithms consist of two things:**
    1. The actions that must be taken, and
    2. The order they must be done in.
- For example, an algorithm to add two numbers:
    1. Take two number inputs
    2. Add numbers using the + operator
    3. Display the result

Algorithm

- **Qualities of Good Algorithms:**
  - Input and output should be defined precisely.
  - Each step in the algorithm should be clear and unambiguous.
  - Algorithms should be most effective among many different ways to solve a problem.
  - An algorithm shouldn't include computer code. Instead, the algorithm should be written in such a way that it can be used in different programming languages.

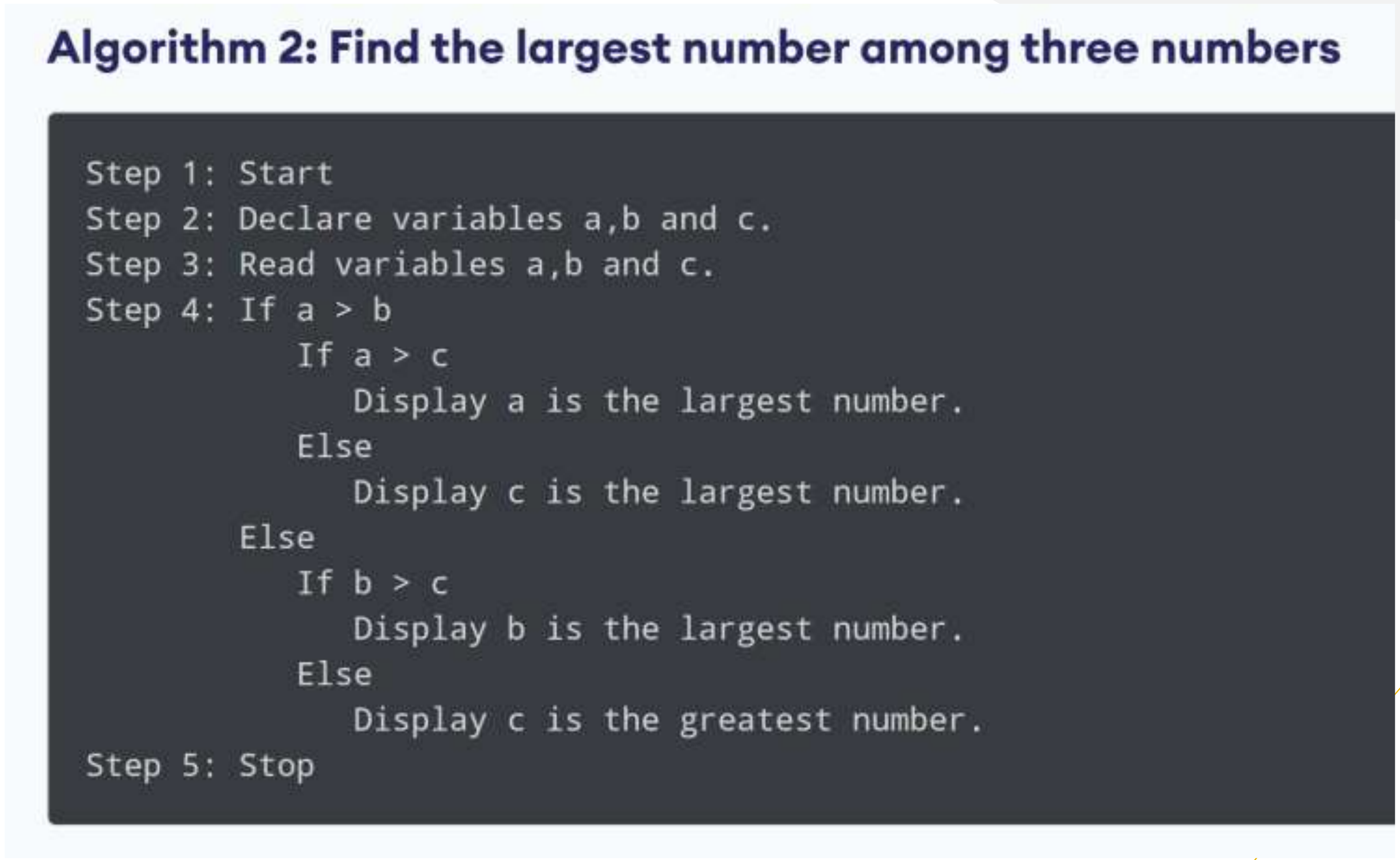
Algorithm



Algorithm 1: Add two numbers entered by the user

```
Step 1: Start
Step 2: Declare variables num1, num2 and sum.
Step 3: Read values num1 and num2.
Step 4: Add num1 and num2 and assign the result to sum.
        sum←num1+num2
Step 5: Display sum
Step 6: Stop
```

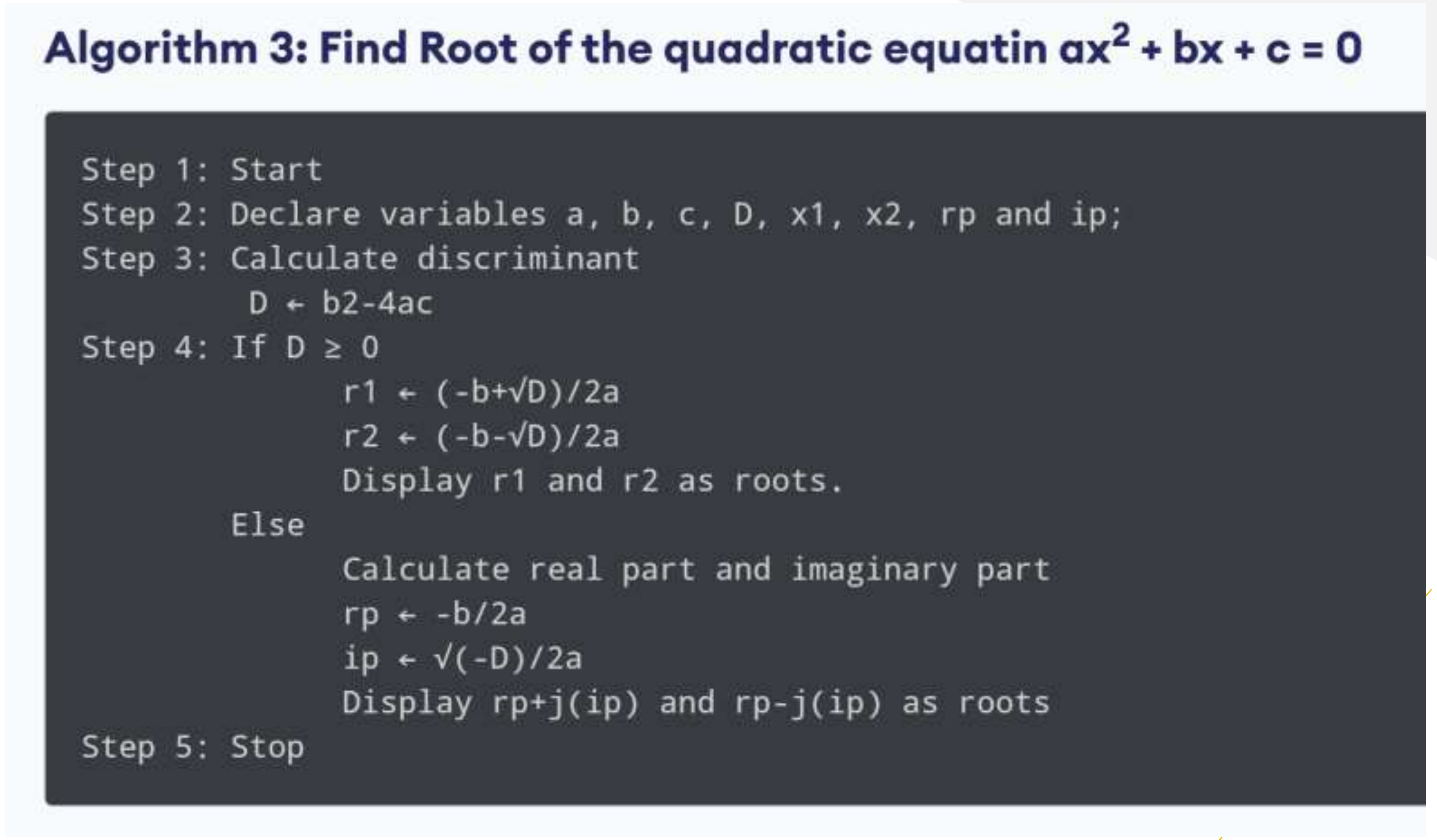
Algorithm



Algorithm 2: Find the largest number among three numbers

```
Step 1: Start
Step 2: Declare variables a,b and c.
Step 3: Read variables a,b and c.
Step 4: If a > b
           If a > c
              Display a is the largest number.
           Else
              Display c is the largest number.
        Else
           If b > c
              Display b is the largest number.
           Else
              Display c is the greatest number.
Step 5: Stop
```

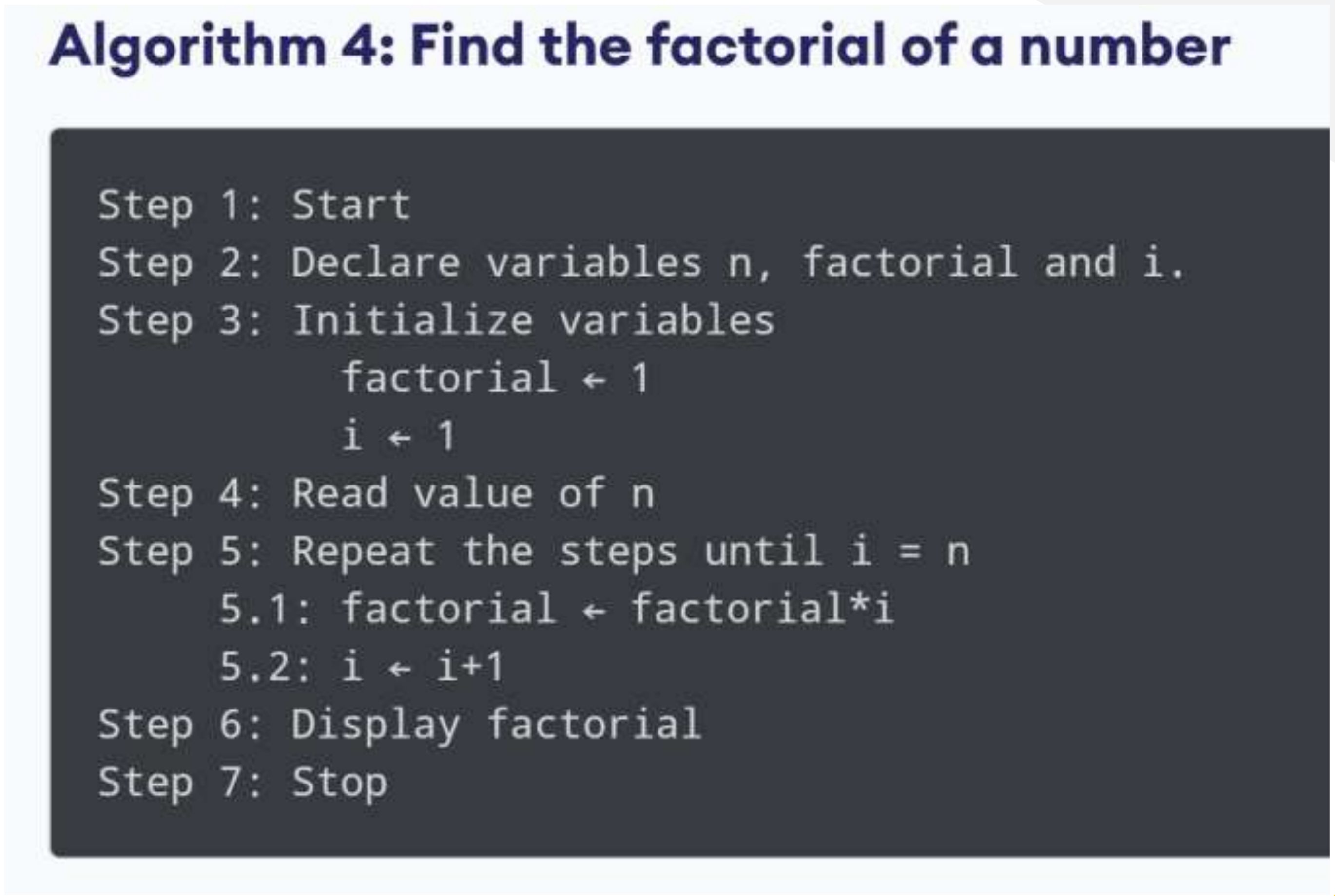
Algorithm



Algorithm 3: Find Root of the quadratic equatin $ax^2 + bx + c = 0$

```
Step 1: Start
Step 2: Declare variables a, b, c, D, x1, x2, rp and ip;
Step 3: Calculate discriminant
         D ← b2-4ac
Step 4: If D ≥ 0
              r1 ← (-b+√D)/2a
              r2 ← (-b-√D)/2a
              Display r1 and r2 as roots.
        Else
              Calculate real part and imaginary part
              rp ← -b/2a
              ip ← √(-D)/2a
              Display rp+j(ip) and rp-j(ip) as roots
Step 5: Stop
```

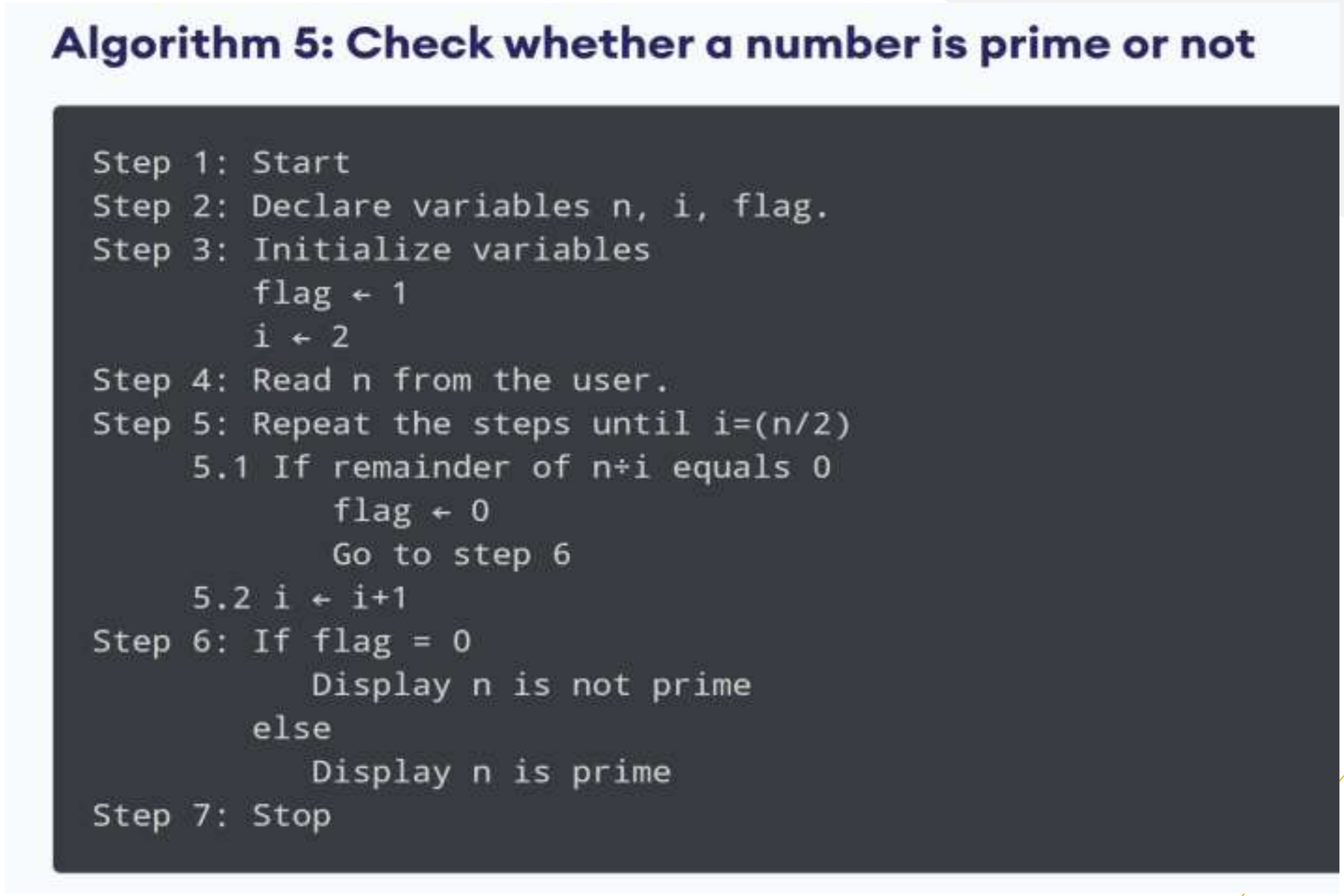
Algorithm



Algorithm 4: Find the factorial of a number

```
Step 1: Start
Step 2: Declare variables n, factorial and i.
Step 3: Initialize variables
          factorial ← 1
          i ← 1
Step 4: Read value of n
Step 5: Repeat the steps until i = n
     5.1: factorial ← factorial*i
     5.2: i ← i+1
Step 6: Display factorial
Step 7: Stop
```

Algorithm



Algorithm 5: Check whether a number is prime or not

```
Step 1: Start
Step 2: Declare variables n, i, flag.
Step 3: Initialize variables
        flag ← 1
        i ← 2
Step 4: Read n from the user.
Step 5: Repeat the steps until i=(n/2)
     5.1 If remainder of n÷i equals 0
            flag ← 0
            Go to step 6
     5.2 i ← i+1
Step 6: If flag = 0
           Display n is not prime
        else
           Display n is prime
Step 7: Stop
```

Types of Algorithms

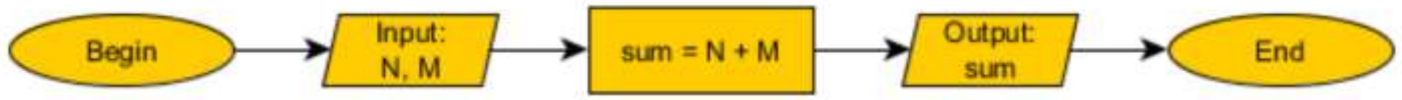
- Algorithms are classified based on the concepts that they use to accomplish a task. While there are many types of algorithms, the **most fundamental types of computer science algorithms** are:
  - **Divide and conquer algorithms** – divide the problem into smaller subproblems of the same type; solve those smaller problems, and combine those solutions to solve the original problem.
  - **Brute force algorithms** – try all possible solutions until a satisfactory solution is found.
  - **Randomized algorithms** – use a random number at least once during the computation to find a solution to the problem.

Types of Algorithms

- **The most fundamental types of computer science algorithms** are:
  - **Greedy algorithms** – find an optimal solution at the local level with the intent of finding an optimal solution for the whole problem.
  - **Recursive algorithms** – solve the lowest and simplest version of a problem to then solve increasingly larger versions of the problem until the solution to the original problem is found.
  - **Backtracking algorithms** – divide the problem into subproblems, each which can be attempted to be solved; however, if the desired solution is not reached, move backwards in the problem until a path is found that moves it forward.
  - **Dynamic programming algorithms** – break a complex problem into a collection of simpler subproblems, then solve each of those subproblems only once, storing their solution for future use instead of re-computing their solutions.

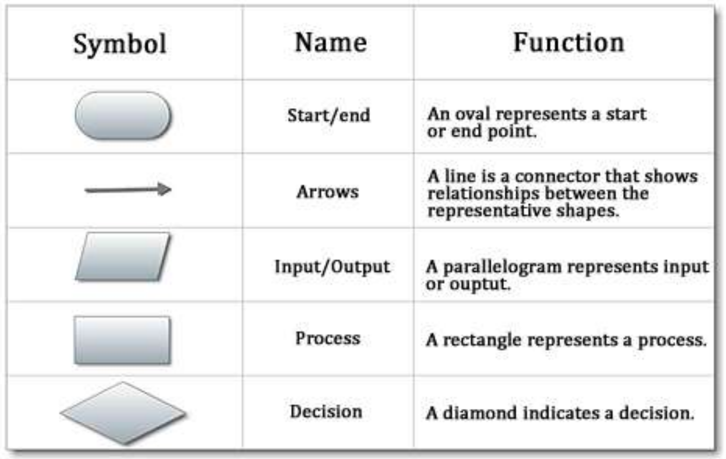
Flowchart

- A ***flowchart*** is a diagrammatic representation of the logic for solving a task. A flowchart is drawn using boxes of different shapes with lines connecting them to show the flow of control.
- The purpose of drawing a flowchart is **to make the logic of the program clearer in a visual form.** The logic of the program is communicated in a much better way using a flowchart.



Flowchart

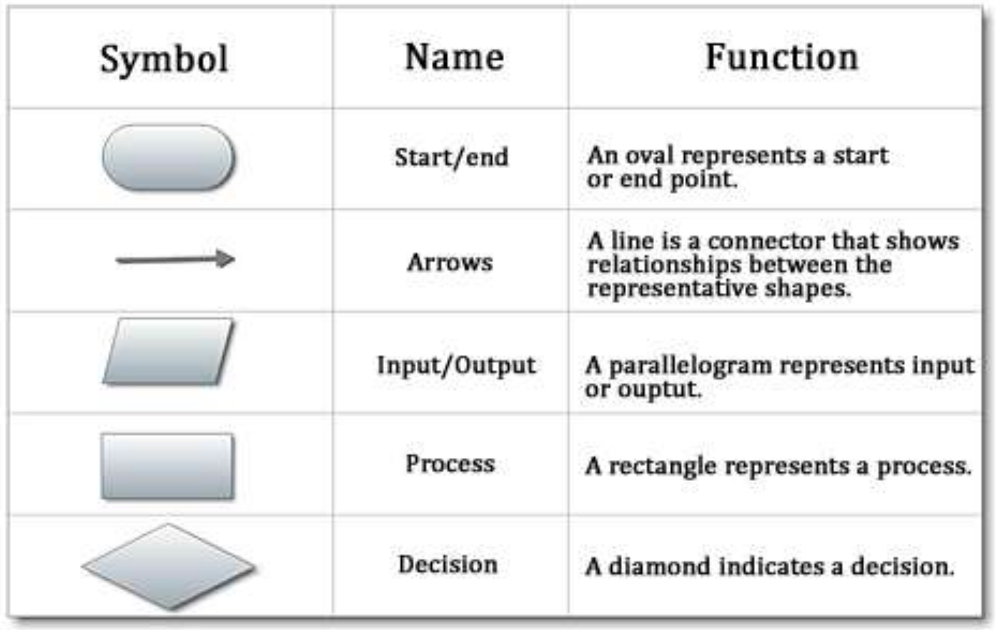
- **Flowchart Symbols:** A flowchart is drawn using different kinds of symbols. A symbol used in a flowchart is for a specific purpose.



Symbol	Name	Function
	Start/end	An oval represents a start or end point.
	Arrows	A line is a connector that shows relationships between the representative shapes.
	Input/Output	A parallelogram represents input or ouptut.
	Process	A rectangle represents a process.
	Decision	A diamond indicates a decision.

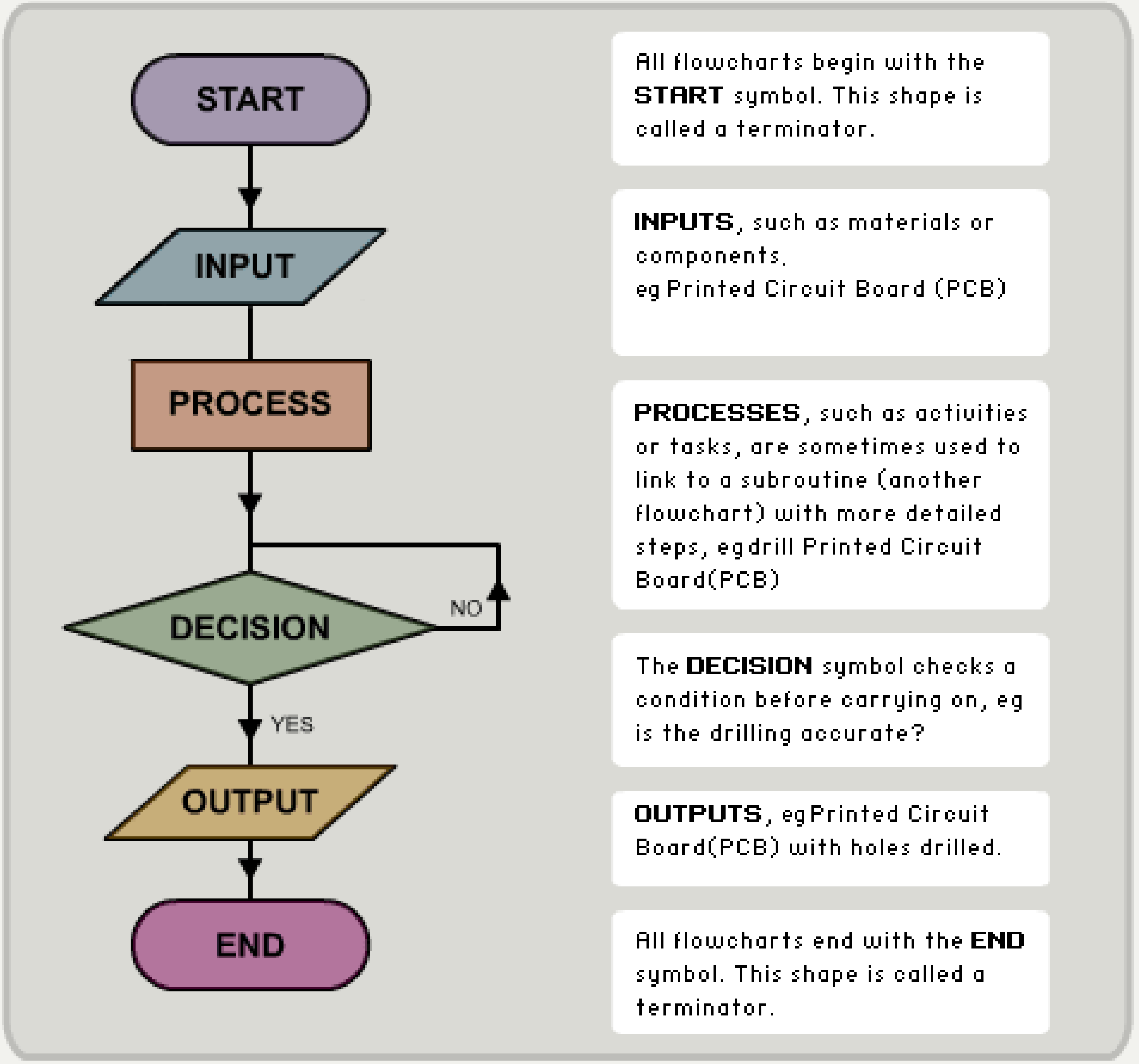
- Every flowchart has to start with a **TERMINAL** shape containing the caption ***START*** and has to end with another TERMINAL shape containing the caption of ***END***.
- **INPUT / OUTPUT** shape is used to indicate the time and place of reading some values and/or giving some output to the user.
- **PROCESS** symbol is used to represent assignments and manipulations of data such as arithmetic operations or movement of data from one variable to the other.

Flowchart



Symbol	Name	Function
⬭	Start/end	An oval represents a start or end point.
→	Arrows	A line is a connector that shows relationships between the representative shapes.
▱	Input/Output	A parallelogram represents input or ouptut.
▭	Process	A rectangle represents a process.
◇	Decision	A diamond indicates a decision.

- **DECISION** symbol represents the comparison of two values. Alternating course of actions will be followed depending on the result of checked criteria.
- **CONNECTOR** symbol is used to represent the exit to or entry from another part of the program. It is used to show the connections of two pages, when your design occupies more then one page.
- **FLOWLINE** symbol is used to show the direction of the program flow between other symbols.



All flowcharts begin with the **START** symbol. This shape is called a terminator.

INPUTS, such as materials or components.
eg Printed Circuit Board (PCB)

PROCESSES, such as activities or tasks, are sometimes used to link to a subroutine (another flowchart) with more detailed steps, eg drill Printed Circuit Board(PCB)

The **DECISION** symbol checks a condition before carrying on, eg is the drilling accurate?

OUTPUTS, eg Printed Circuit Board(PCB) with holes drilled.

All flowcharts end with the **END** symbol. This shape is called a terminator.

Flowchart

- Fig: Sample flowchart of a program.

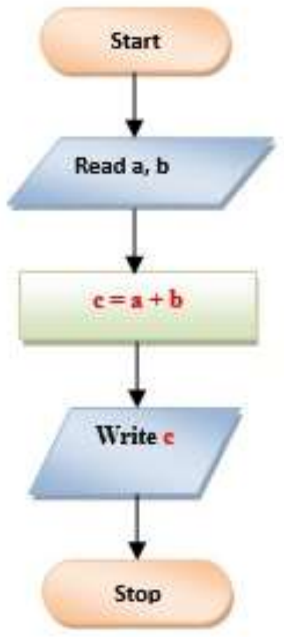
Flowchart

- To find sum of two numbers:

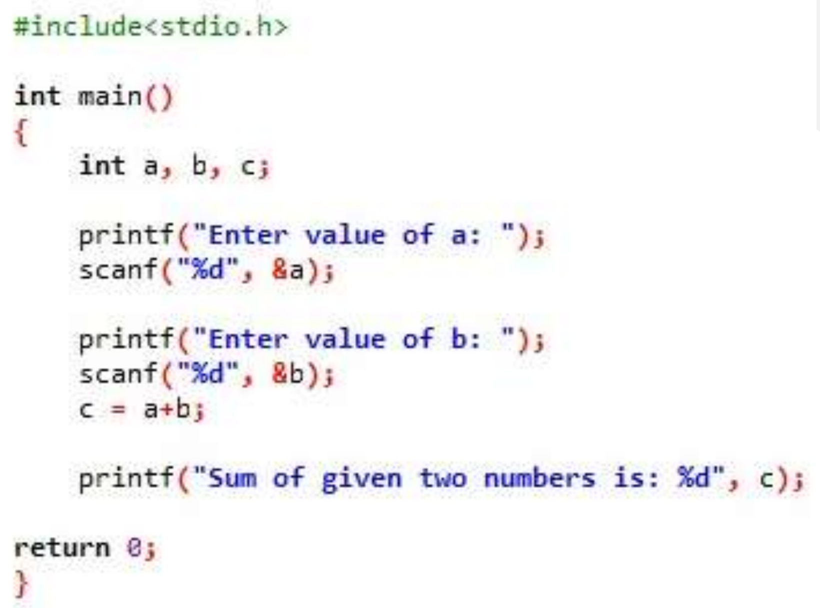
Algorithm

1. Start
2. Read a, b
3. c = a + b
4. Print or display c
5. Stop

Flowchart



C Program



```
#include<stdio.h>

int main()
{
    int a, b, c;

    printf("Enter value of a: ");
    scanf("%d", &a);

    printf("Enter value of b: ");
    scanf("%d", &b);
    c = a+b;

    printf("Sum of given two numbers is: %d", c);

return 0;
}
```

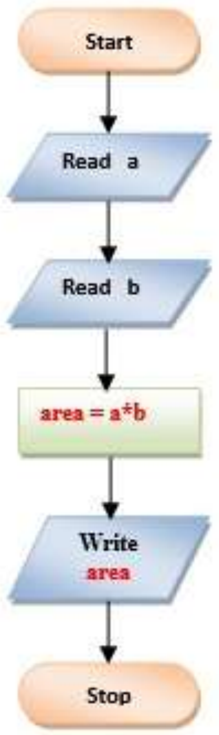
Flowchart

- To find area of a rectangle:

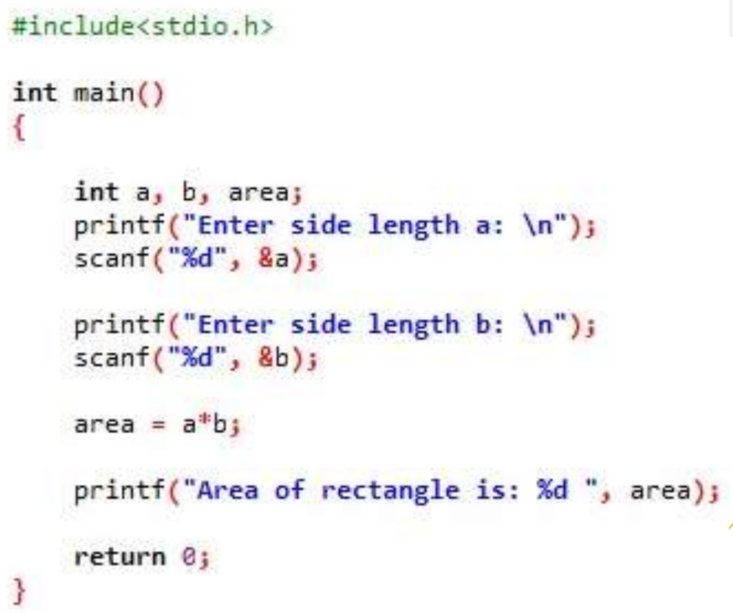
Algorithm

1. Start
2. Read side length, a
3. Read side length b
4. area = a*b
5. Print or display area
6. Stop

Flowchart



C Program

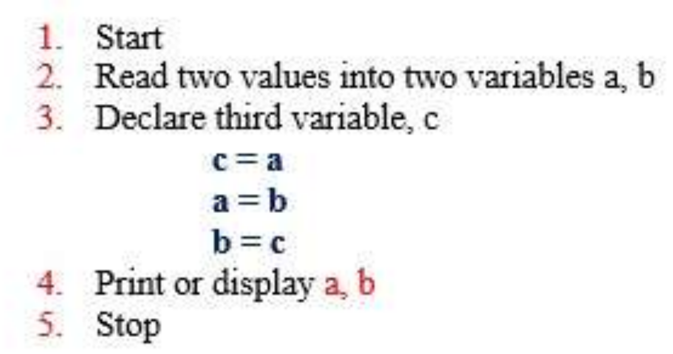


```
#include<stdio.h>

int main()
{

    int a, b, area;
    printf("Enter side length a: \n");
    scanf("%d", &a);

    printf("Enter side length b: \n");
    scanf("%d", &b);

    area = a*b;

    printf("Area of rectangle is: %d ", area);

    return 0;
}
```

Flowchart

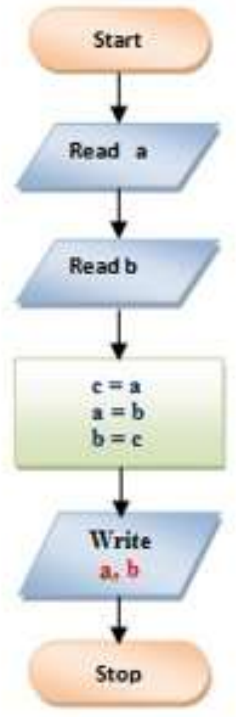
- To interchange the value of two numbers :

Algorithm

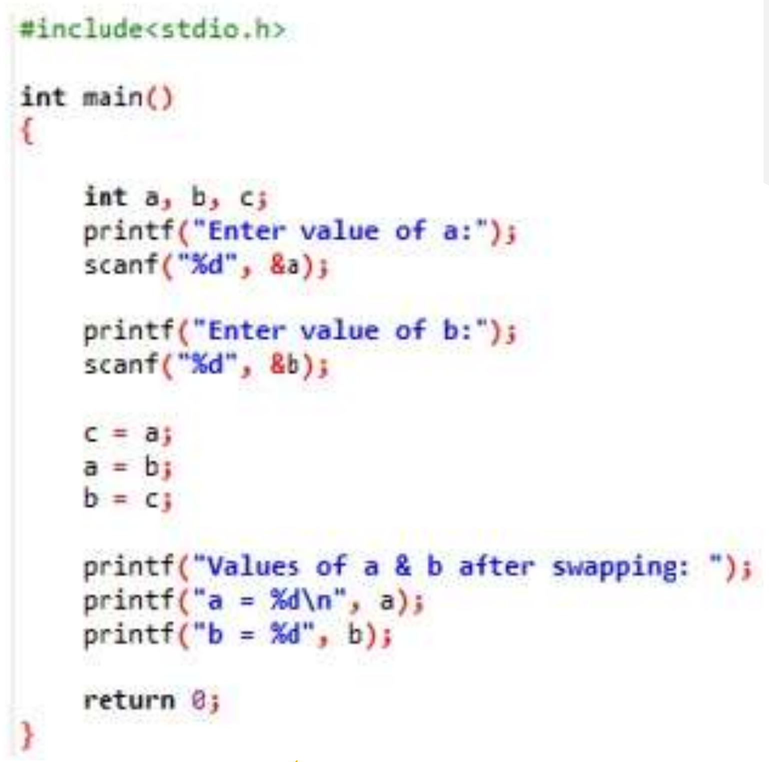


```
1.  Start
2.  Read two values into two variables a, b
3.  Declare third variable, c
            c = a
            a = b
            b = c
4.  Print or display a, b
5.  Stop
```

Flowchart



C Program

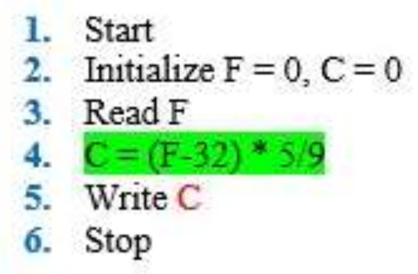


```c
#include<stdio.h>

int main()
{

    int a, b, c;
    printf("Enter value of a:");
    scanf("%d", &a);

    printf("Enter value of b:");
    scanf("%d", &b);

    c = a;
    a = b;
    b = c;

    printf("Values of a & b after swapping: ");
    printf("a = %d\n", a);
    printf("b = %d", b);

    return 0;
}
```

Flowchart

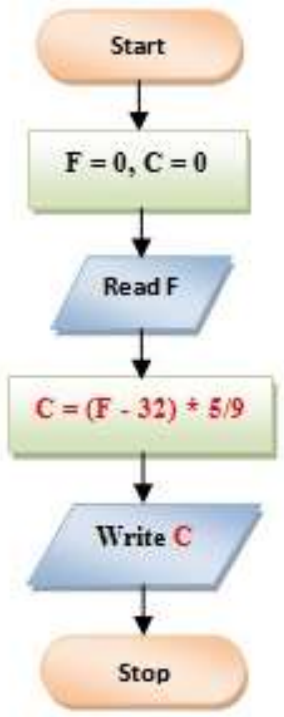
- To convert temperature from Fahrenheit to Celsius:

Algorithm

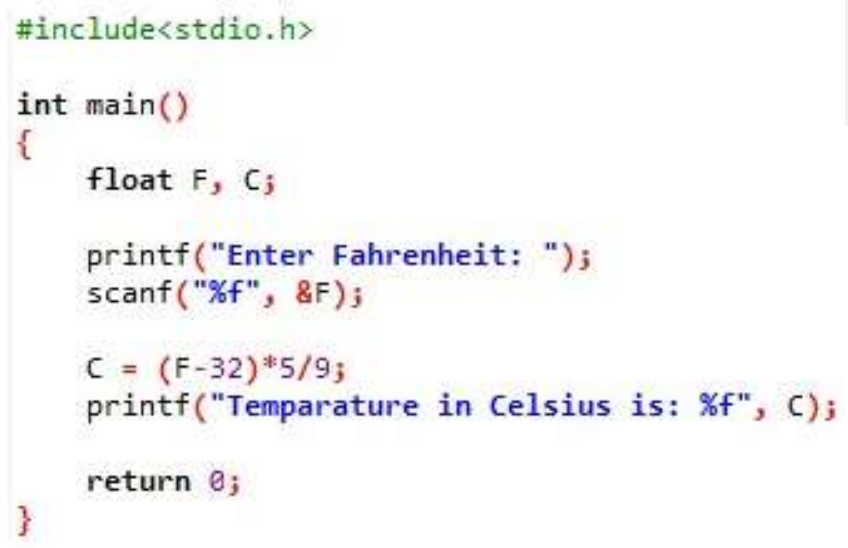


1. Start
2. Initialize F = 0, C = 0
3. Read F
4. C = (F-32) * 5/9
5. Write C
6. Stop

Flowchart



C Program

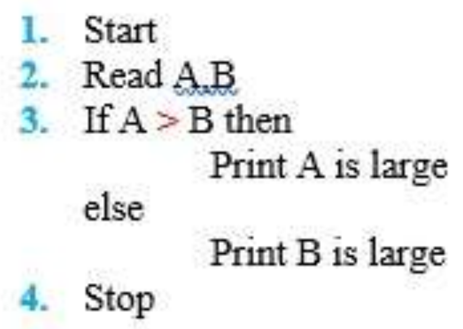


```
#include<stdio.h>

int main()
{
    float F, C;

    printf("Enter Fahrenheit: ");
    scanf("%f", &F);

    C = (F-32)*5/9;
    printf("Temparature in Celsius is: %f", C);

    return 0;
}
```

Flowchart

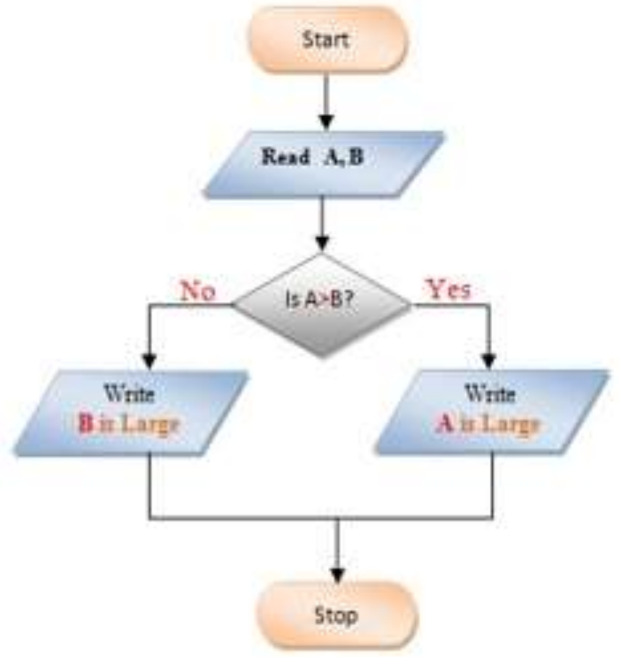
- To find the greatest of two numbers :

Algorithm

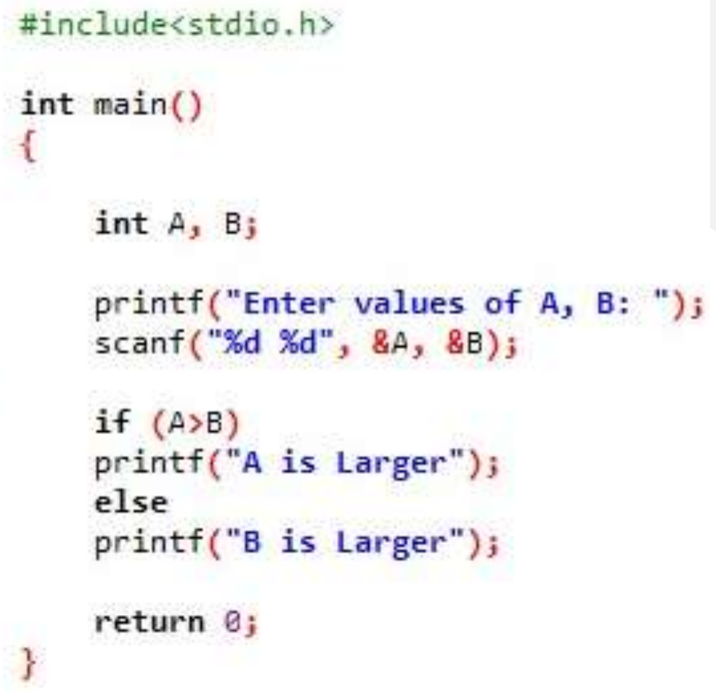


```
1. Start
2. Read A,B
3. If A > B then
            Print A is large
   else
            Print B is large
4. Stop
```

Flowchart



C Program

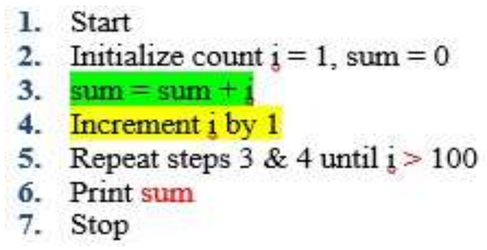


```
#include<stdio.h>

int main()
{

    int A, B;

    printf("Enter values of A, B: ");
    scanf("%d %d", &A, &B);

    if (A>B)
    printf("A is Larger");
    else
    printf("B is Larger");

    return 0;
}
```

Flowchart

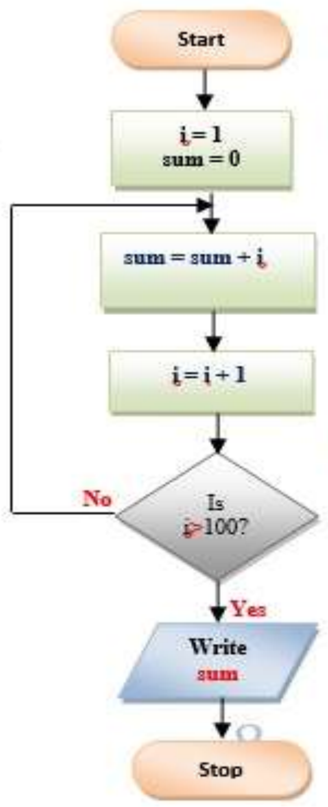
- To compute the sum of integers 1 to 100:

Algorithm

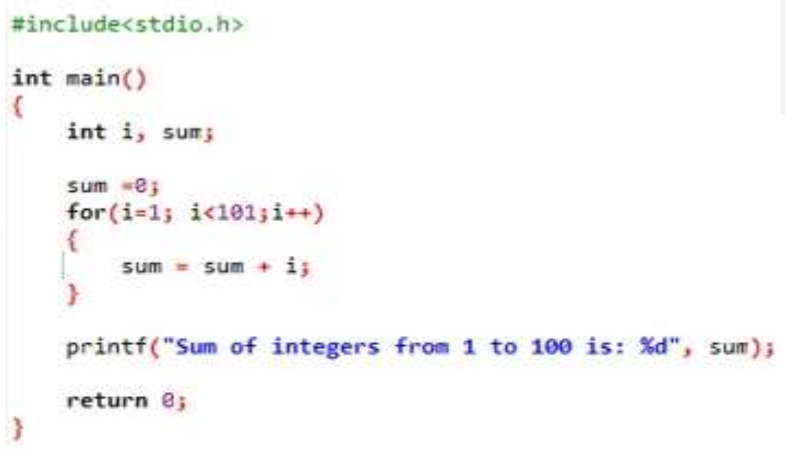


1. Start
2. Initialize count i = 1, sum = 0
3. sum = sum + i
4. Increment i by 1
5. Repeat steps 3 & 4 until i > 100
6. Print sum
7. Stop

Flowchart



C Program



```
#include<stdio.h>

int main()
{
    int i, sum;

    sum =0;
    for(i=1; i<101;i++)
    {
        sum = sum + i;
    }

    printf("Sum of integers from 1 to 100 is: %d", sum);

    return 0;
}
```

? THE END