Week# 4
Arithmetic Operators & Mathematical Expressions

Dr Taimur Ahmed

Department of IT & CS, PAF-IAST

Recapitulation
Problem solving techniques

Recap – Algorithms and Flow charts

- **Algorithm** or **Pseudocode**
  - *Algorithm is a step-by-step procedure, which defines a set of instructions to be executed in a certain order to get the desired output*
- Algorithms help to **understand the problem** and define our approach to solve the problem in our program

Problem − Design an algorithm to add two numbers and display the result

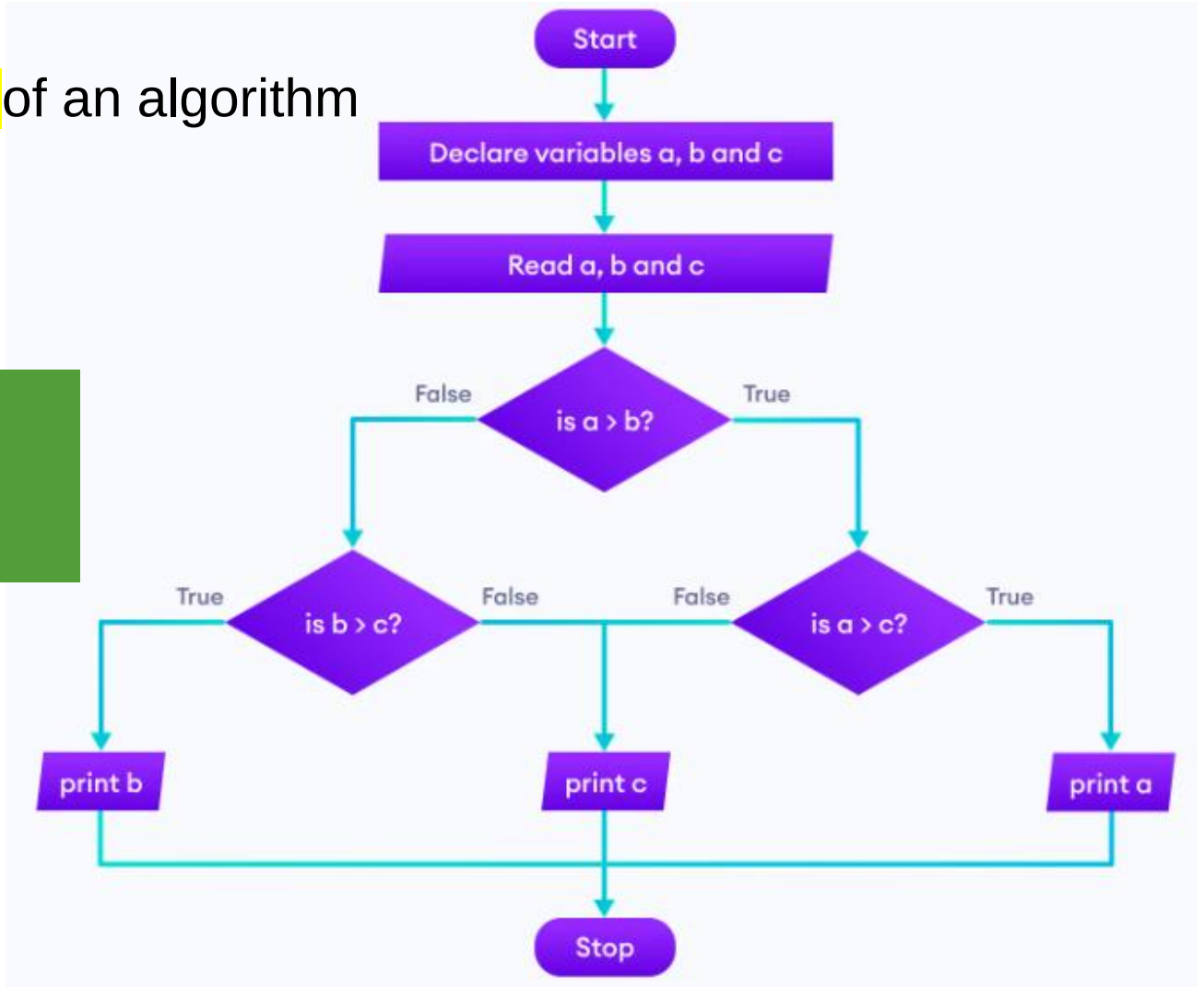
Step 1: START
Step 2: declare three integers **a, b & c**
Step 3: define values of **a & b**
Step 4: add values of **a & b**
Step 5: store output of **Step 4** in **c**
Step 6: print **c**
Step 7: STOP

Recap – Algorithms and Flowcharts

❑ Flowchart

- ➢ Flowchart is a **diagram representation** of an algorithm

Problem – Find the largest among three different numbers entered by the user



Recap – Variables & Data types

- Variable is a **storage location** in memory with a give name
  - Name of a variable should follow certain rules
  - Variable must be defined before it is used in program
- Data type defines **nature of data**

Data types	Keyword
Boolean	`bool`
Character	`char`
Integer	`int`
Floating point	`float`
Double floating point	`double`
Valueless	`void`
Wide character	`wchar_t`

Lecture# 6
Operators

Types of Operators

- Operator is a **symbol** that tells the compiler to perform specific **mathematical or logical manipulations**
- C++ has an extensive range of operators and provides following types
  - Arithmetic Operators
  - Relational Operators
  - Logical Operators
  - Bitwise Operators
  - Assignment Operators
  - Misc Operators

Arithmetic Operators

- Arithmetic operators allow numeric/mathematical calculations
- Types of arithmetic operators
  - Unary arithmetic operators (1 operand) `-5`
  - Binary (2 operands) `13 - 7`
    - More discussion is in next slides
  - Ternary (3 operands) `exp1 ? exp2 : exp3`
    - Ternary operator allows to execute different code depending on the value of a condition

```
<condition> ? <true-case-code> : <false-case-code>;

int five_divided_by_x = ( x != 0 ? 5 / x : 0 );
```

Unary Arithmetic Operators

- **Unary operators** require only **one operand**

```
y = +x;          // same as y = +1 * x
y = -x;          // same as y = -1 * x
y = ++x;         // same as y = x + 1
y = --x;         // same as y = x – 1
y = x++;         // same as y = x then x + 1
y = x--;         // same as y = x then x - 1
```

SYMBOL	OPERATION	EXAMPLE	RESULT
++	increment	`int i = 0; ++i`	1
--	decrement	`int i = 9; --i`	8

Unary Arithmetic Operators

```cpp
/* Lecture# 6
   Example code for Unary operators
*/

#include <iostream>
using namespace std;

int main()
{
    int y, x = 10;      // variable defination and initialization

    y = x;              // our instruction

    cout << "The value of y is" << '\t' << y << endl;       // value of x

    cout << "The value of y after ++y is" << '\t' << ++y << endl;       // increment operator

    cout << "The value of y after --y is" << '\t' << --y << endl;       // decrement operator

    cout << "The value of y after y-- is" << '\t' << y-- << endl;       // first print y and then decrement

    cout << "Now the value of y after y-- is" << '\t' << y << endl;     // print y

    cout << "The value of y after y++ is" << '\t' << y++ << endl;       // first print y and then increment

    cout << "Finally the value of y after y++ is" << '\t' << y << endl; // print y

   }
```

Unary Arithmetic Operators

Program Output

```
The value of y is       10
The value of y after ++y is     11
The value of y after --y is     10
The value of y after y-- is     10
Now the value of y after y-- is 9
The value of y after y++ is     9
Finally, the value of y after y++ is     10
```

Binary Arithmetic Operators

SYMBOL	OPERATION	EXAMPLE	RESULT
+	addition	`ans = 7 + 3;`	10
-	subtraction	`ans = 7 - 3;`	4
*	multiplication	`ans = 7 * 3;`	21
/	division	`ans = 7 / 3;`	2
%	modulus	`ans = 7 % 3;`	1

Binary Arithmetic Operators – A closer look

- / (division) operator performs **integer division** **if both operands are integers**

```
cout << 13 / 5;      // displays 2
cout << 13 / 5.0;    // displays 2.6
```

- If either operand is **floating point**, the result is floating point

Binary Arithmetic Operators – A closer look

- % (modulus) operator computes the **remainder** resulting from integer division

```
cout << 13 % 5;   // displays 3
cout << 13 % 5.0; // error
```

- % requires **integers for both operands**

Binary Arithmetic Operators

```
1  /* This program calculates weekly wages, including overtime.
2     Example program for Arithmetic Operator.
3     Note multi-line comment.
4  */
5
6  #include <iostream>
7  using namespace std;
8
9  int main()
10 {
11     double regularWages,     // variable to hold value of regular wages
12     basePayRate = 20,        // Basic pay rate per hour
13     regularHours = 40,       // Hours worked in week
14     overTimeWages,           // holds overtime wages
15     overTimePayRate = 30,    // overtime pay rate per hour
16     overTimeHours = 10,      // overtime hours worked
17     totalWages;              // holds value of total wages
18
19     // Calculate regular wages
20
21     regularWages = basePayRate * regularHours;      // multiplication operator
22
23     // Calculate overtime wages
24
25     overTimeWages = overTimePayRate * overTimeHours;// multiplicaiton operator
26
27     // Calculate total wages
28
29     totalWages = regularWages + overTimeWages;      // addition operator
30
31     // Print the total wages
32
33     cout << "Wages for this week are $" << totalWages << "." << endl;
34
35     return 0;
36 }
```

Program output

```
Wages for this week are $1100.
```

Comments
Single line, multi-line comments

Comment

- Used to **document** parts of the program
- Intended for person reading the source code of the program:
  - Indicate the purpose of the program
  - Describe the use of variables
  - Explain complex sections of code
- Are **ignored by the compiler**
- As a good programming practice **use comments generously** in your code

Comment – Single line

- Begin with // through to the end of line:

```
int length = 12; // length in inches
int width = 15;  // width in inches
int area;        // calculated area

// calculate rectangle area
area = length * width;
```

Comment – Multi line

- Begins with /* and ends with */
- Can span multiple lines:

```
/* this is a multi-line
   comment
*/
```

- Can begin and end on the same line:

```
int area;   /* calculated area */
```