EE058IU -Programming For Engineers Laboratory

Lab 1 – Part A

Variable - Data Types

Lab 1 – Part B

Making Decisions

Full name + Student ID: ………………………………………………………………………………………………

………………………………………………………………………………………………

Class: ……

Group: ………

Date: ……

I. Objectives

1. Write simple computer programs in C language, use the input/output statements, understand the fundamental data types and learn computer memory concepts.
2. Use assignment, arithmetic, relational and logical operators, learn the precedence order of these operators and write simple decision-making / branching statements.
3. Use basic problem-solving techniques, develop algorithms through the process of top-down, stepwise refinement, use the **if**-selection statement, the **if…else…**-selection statement and **switch**-statement to select actions.

II. Pre-Lab Preparation

Students are required to review the theory of the topics before the lab time.

III. In-Lab Procedure

Exercise 1

Write a C program to determines whether an input number is even or odd.

Output:

```
Input an integer: 15
15 is an odd integer
```

Exercise 2

Write a C program to check whether an input year is a leap year or not.

Provide a **flow chart** before writing the program.

Output:

```
Input a year :2024
2024 is a leap year.
```

Exercise 3

Write a C program to identify whether a triangle is Equilateral, Isosceles or Scalene based on the given sides.

Example output:

Input three sides of triangle: 30 30 50

This is an isosceles triangle.

Exercise 4

Write a C program to input radius of a circle from user and find diameter, circumference, and area of the circle. User can decide what unit should be displayed.

For example:

<u>Output:</u>

```
Enter measurement unit: cm
Enter the radius of a circle: 3
Diameter of circle = 6.00 cm
Circumference of circle = 18.84 cm
Area of circle = 28.26 sq. cm
```

Exercise 5

Write a C program that calculate the roots of the second order equation:

$$ax^2 + bx + c = 0$$

Display a proper message if the equation has no real root.

Hints: Use include<math.h>

<u>Output example:</u>

Enter the Values of a : 1

Enter the Values of b : -4

Enter the Values of c : 3

The roots of equation are

First Root : 3.00

Second Root : 1.00

Exercise 6

Write a C program that calculate the total payment of Taxi service, in which:

- The first kilometer: 10,000 VND;
- The travelled distance is between 1 km and 30 km: 5,000 VND/km;
- The travelled distance is greater than 30 km: 3,000 VND/km;

Example

<u>Output:</u>

Enter total distance in km: 5.2

The total cost that a passenger has to pay: 31,000 VND

Exercise 7

Write a C program to read student's name, student's ID and marks of three subjects (Literature, Math and English) and calculate the total, average and division.

Division is defined as follows:

Average $\geq$ 60: **A**

48 $\leq$ Average $<$ 60: **B**

36 $\leq$ Average $<$ 48: **Pass**

Average < 36: **Fail**

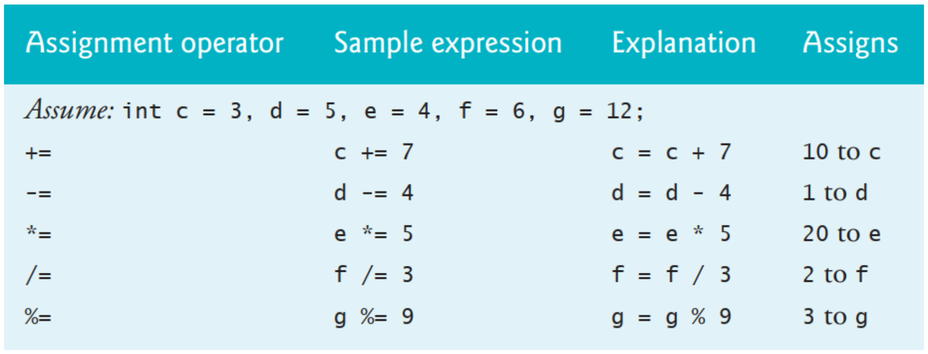
For Example

Output:

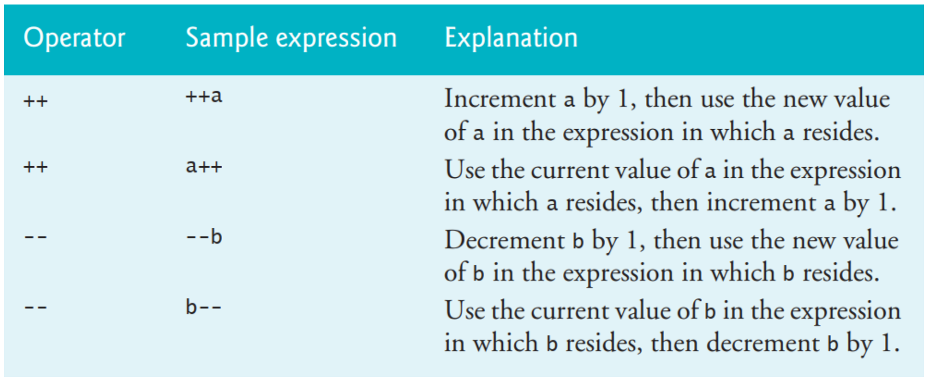
```
Input the Name of the Student: Kien
Input the No of the student: 10
Input the marks of Literature, Math and English: 50 80 70

Name of Student: Kien
ID: 10
Marks in Literature: 50
Marks in Math: 80
Marks in English: 70
Total Marks = 200
Average = 66.67
Division: A
```

THE END

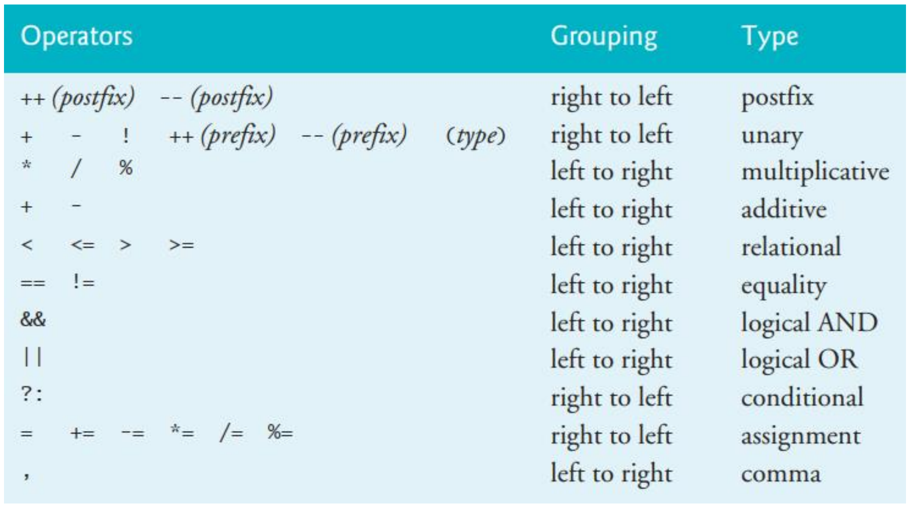
Useful Operators



Assignment operator	Sample expression	Explanation	Assigns
Assume: `int c = 3, d = 5, e = 4, f = 6, g = 12;`			
`+=`	`c += 7`	`c = c + 7`	10 to `c`
`-=`	`d -= 4`	`d = d - 4`	1 to `d`
`*=`	`e *= 5`	`e = e * 5`	20 to `e`
`/=`	`f /= 3`	`f = f / 3`	2 to `f`
`%=`	`g %= 9`	`g = g % 9`	3 to `g`



Operator	Sample expression	Explanation
`++`	`++a`	Increment `a` by 1, then use the new value of `a` in the expression in which `a` resides.
`++`	`a++`	Use the current value of `a` in the expression in which `a` resides, then increment `a` by 1.
`--`	`--b`	Decrement `b` by 1, then use the new value of `b` in the expression in which `b` resides.
`--`	`b--`	Use the current value of `b` in the expression in which `b` resides, then decrement `b` by 1.



Operators	Grouping	Type
`++` *(postfix)* `--` *(postfix)*	right to left	postfix
`+` `-` `!` `++` *(prefix)* `--` *(prefix)* *(type)*	right to left	unary
`*` `/` `%`	left to right	multiplicative
`+` `-`	left to right	additive
`<` `<=` `>` `>=`	left to right	relational
`==` `!=`	left to right	equality
`&&`	left to right	logical AND
`||`	left to right	logical OR
`?:`	right to left	conditional
`=` `+=` `-=` `*=` `/=` `%=`	right to left	assignment
`,`	left to right	comma