## WEEK 4: MORE ON ELEMENTS OF C++ AND DATA TYPES, OPERATORS, TYPES OF EXPRESSIONS AND INPUT/OUTPUT (I/O)

**Objectives**

Following completion of this topic, students should be able to:

- Understand the C++ arithmetic operators and their order of evaluation in mixed expressions
- Output data in predefined formats
- Write programs in the recommended coding style

---

### TASK 0. DATA VALUES AND ARITHMETIC OPERATIONS:

*NOTE: This task is to be completed in your own time **before** the start of your lab.*
*This task is a pen-and-paper task – no computer is required.*

Recall that there are four basic data types used in C++: integers, floating-point numbers, character values and Boolean types. Answer the following question.

**a.** Write the following decimal numbers using C++ exponential notation where there is only one non-zero digit before the decimal point:
Example: 245.6 = 2.456 x $10^2$ = 2.456e2

i. 126. =
ii. 3426.78 =
iii. .341 =
iv. .00926 =

Each of the C++ mathematical operators follows an order of precedence as shown in the following table.

| Operator | Meaning | Type |
|---|---|---|
| + - | Positive / Negative | Unary |
| * / % | Multiplication / Division / Modulus | Binary |
| + - | Addition / Subtraction | Binary |

All of the mathematical operators, except modulus (%), may be used in expressions with any numerical type as the operands.

- The modulus (%) is only applicable to integer expressions. If you attempt to use it in an expression with a non-integer type as an operand, the compiler will produce an error. When performing modulus on a number, the result is the remainder portion of a divide expression. For example:

  7 % 2 = 1

- A divide operation which contains **both** operands of the integer type, ensures an 'integer division' is executed. The result is the quotient portion of a divide expression. For example:

  7 / 2 = 3
- Any expression, which contains mixed types as operands, will have the one side promoted to the type of the other before the expression is evaluated. Integers will be promoted to floats/doubles and floats will be promoted to doubles and the result will be of that same type.

  For example:

  7.0 / 2

  will cause the 2 to be promoted to a double and the following expression will be evaluated.

  7.0 / 2.0 = 3.5

**b.** Using parentheses, rewrite the following C++ expressions to correctly indicate their order of evaluation considering the operations *precedence* rules. Then determine the value of each expression, showing all the steps leading to the result:

i. 2.0 + 3.0 / 12.0 * 8.0 / 4.0 =

ii. 2.0 * 5.0 + 7.0 =

iii. 1.0 + 23.0 / 10.0 =

iv. 3.0 + 5.0 * 6.0 =

v. 10.0 – 2.0 / 6.0 + 3.0 =

vi. 10.0 – 2.0 / 6 + 3.0 =

vii. 10 * ( 1 + 7 % 3) =

viii. 10 * ( 1 + 7.0 % 3) =

ix. 2 * 3 / 12 * 8 / 4 =

x. 2 * 5 + 7 =

xi. 1 + 23 / 10 =

## TASK 1. MATHEMATICAL OPERATIONS

a. Create a directory for this week's lab work and move to it.

b. Create a C++ program named *'task1.cpp'* which contains the expressions in **Task 0** question **b** in `cout` statements to verify your answers.

c. Compile and run the program.

d. Compare the output with the answers you calculated in **Task 0** question **c.** If there are any discrepancies, check your answers to determine the errors. Be sure that you understand how the operations have been performed.

## TASK 2. INCREMENT/DECREMENT OPERATIONS

The increment and decrement operators increase/decrease a variable's value by **one** and can be applied to any of the basic types. They take two forms.

(i) Pre- increment/decrement – Increment/decrement the variable **before** usage.

(ii) Post- increment/decrement – Increment/decrement the variable **after** usage.

For example:

```
int i = 1;
cout << i++ << endl;
```

will produce output of `1`. As the operation includes a post-increment, `a` is incremented **after** its value has been applied to the `cout` statement.
After the `cout` statement the value of `i` will be `2`.
However

```
int i = 1;
cout << ++i << endl;
```

will produce output of `2`. In this case, `a` is incremented **before** it is used in the `cout` statement. The value of `i` after output is the same for the decrement operator.

**a.** What will be produced when the following code segment is executed

```
char a = 'a', z = 'Z';
cout << a << " " << ++a << " " << a++ << endl;
cout << z << " " << z-- << " " << --z << endl;
```

**b.** What are the values stored in the two variables, `a` and `z`, after the `cout` statements have executed?

**c.** Incorporate the code in step **a** into a C++ program named *'task2.cpp'* to test your answers from steps **a** and **b**.

## TASK 3. INPUT OPERATIONS USING THE STREAM EXTRACTION OPERATOR

The stream extraction operator '>>' allows data to be extracted from the input stream and stored in variables.

- Input is delimited by white-space characters (new-line, space, tab, etc). That is, it uses white-space to indicate the end of each type input.

- Preceding white-space characters are also ignored.

It behaves according to the type of the variable. For example;

- If the input variable is of type `char`, after ignoring preceding white-space characters, **one** character will be extracted from the stream and stored in the variable. Any input whatsoever (except white-space) can be stored in a `char` type (using >>).
- If the input variable is of type `int`, after ignoring preceding white-space characters, all numerical characters (0-9) will be extracted, converted to an integer value and stored in the variable. Leading + or – characters will determine whether the value is positive or negative. Extraction will cease when a non-digit is encountered.
  - If a non-numerical character is encountered **before** a numerical character, an input error will result (not reported, of course), and no value will be stored.
- If the input variable is of type `float/double`, after ignoring preceding white-space characters, all characters, which represent a floating-point number, will be extracted, converted to a floating-point value and stored in the variable. The input stops when a character which does not fit the allowable patterns for a float is encountered.
  - If the input does not represent a valid floating-point number, an input error will result and, as above, no value is stored.

  The following are examples of valid input values for this type.
  - 1
  - 1.
  - 1.1
  - .1
  - 1e5
  - -1.0E-3

**a.** Consider the following code segment. ***Note: Do not implement it yet.***

```
int i = 0;
float f = 0;
char ch = 'a';
cin >> i >> ch >> f;
cout << "i: " << i << "\nch: " << ch << "\nf: "
         << f << endl;
```

**b.** What will be the output when the input is:

1. `1 A 45`
2. `1A45`
3. `1 9 45`
4. `1 945`
5. `B 45.6`
6. `1BC5.6`

**c.** Incorporate the code from step **a** into a C++ program named *'task3.cpp'*.

**d.** Compile and run the program, providing as input the values from step **b**.

**e.** Compare the output with your written answers. Where your answers differ, try to understand where you went wrong before continuing.

## TASK 4. INPUT OPERATIONS USING THE GET FUNCTION

While the stream extraction operator ignores white-space, sometimes it is necessary to extract white-space from the stream rather than merely ignoring it. There are pre-defined functions that accomplish this task. The `get` function is one. It extracts one character from the input stream and returns its value. Assuming a variable, `ch`, of type `char` has been declared, the following code,

```
cin.get(ch);
```

extracts the next character from the input stream and assigns its value to the variable, `ch`.

***a.*** Consider the following code segment. ***Note: Do not implement it yet.***

```
int i = 0;
float f = 0;
char ch = 'a';
cin >> i;
cin.get(ch);
cin >> f;
cout << "i: " << i << "\nch: " << ch << "\nf: "
         << f << endl;
```

b. What will be the output when the input is:

| | |
|---|---|
| 1. `1 A 45` | 4. `1 945` |
| 2. `1A45` | 5. `B 45.6` |
| 3. `1 9 45` | 6. `1BC5.6` |

c. Incorporate the code from step **a** into a C++ program named *'task4.cpp'*.

d. Compile and run the program, providing as input the values from step **b**.

e. Compare the output with your written answers. Where your answers differ, try to understand where you went wrong before continuing.

## TASK 5. CHALLENGE TASK

Write a program that determines the letter that lies halfway between two letters of the alphabet as input by the user. For example, if the user inputs 'A' and 'Z', the output should be 'M'. Do you know how to deal with upper vs. lower case letters? Does it matter if the letters are input in reverse order?

**SUBMISSION**

a. Show the files '*task1.cpp*', '*task2.cpp*', '*task3.cpp*', and '*task4.cpp*' to your teacher or classroom tutor by the end of Week 4 (Friday Practicum), online student use screen share on zoom.