Programming Technique

Lab 4 – Loop structures

1 Expected outcomes

- Understand loop structures and be able to use them to solve required problems.
- Learn how to solve problems by decomposing them into smaller ones with repeating tasks.

2 Mandatory exercises

Exercise 1. Write a program to print the ASCII values of letters and numbers (excluding special symbols).

Guidelines:

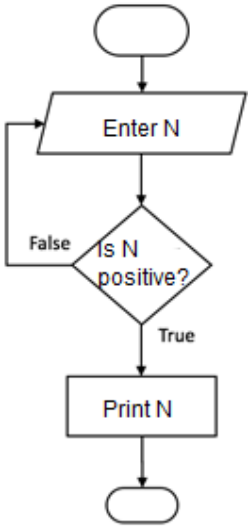
a) Data: The program should have a variable to store the character to be printed. Its data type should be `char`.
b) Algorithm:
   - Use loop structure to cover all of the following cases:
     - Number characters: '0' – '9'.
     - Lower-case letters: 'a' – 'z'.
     - Upper-case letters: 'A' – 'Z'.
   - Learn how to typecast to get the ASCII value of the above characters.

Exercise 2. Write a program that allows the user to:

a) Enter a **positive integer** N, only stops when the user enters the correct number.
b) Print N.

Guidelines:

a) Data: N is a positive integer, the data type can be unsigned long.
b) Algorithm:



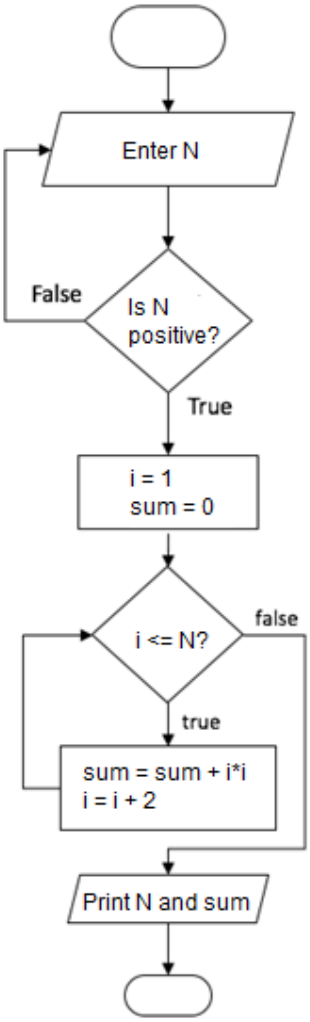
- Because user needs to enter N **before** the program performs checking → we should use do...while.
- The check to see if N is positive is the exiting condition.

Exercise 3. Write a program that allows the user to:

a) Enter a positive integer N, only stops when the user enters the correct number.
b) Compute the **sum of squares** of all **odd** numbers from 1 to N using loop structure.
c) Print N and the result – the printing format should be aesthetic.

Guidelines:

a) Data: the program should have the following variables:
   - A number to store a positive integer N: you can use unsigned long.
   - A number to store the sum of squares: it can be unsigned long long.
b) Algorithm:



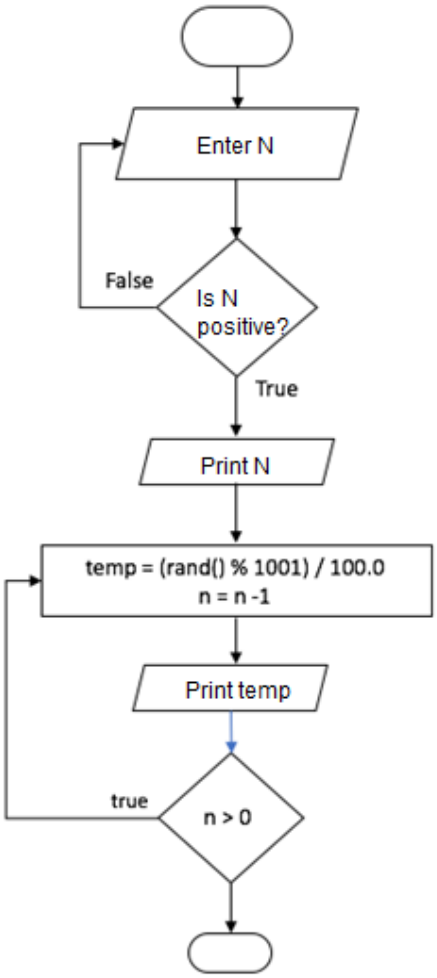
- When calculating the sum, because the number of loops is already known → we can use `for` structure.

Exercise 4. Write a program that allows the user to:

a) Enter a positive integer N, only stops when the user enters the correct number.
b) Print N random real-valued numbers in decimal.
c) Print N and the results – the printing format should be aesthetic.

Guidelines:

a) Data:
   - N is positive: you can use unsigned long.
   - A variable to store a real-valued number in decimal: you can use float.
b) Algorithm:



- Refer to the `rand()` function in C++ to generate a random positive integer with value from 0 to `RAND_MAX`.

Exercise 5. Write a program that allows the user to:

a) Enter a positive integer N, only stops when the user enters the correct number.
b) Generate N random real-valued numbers in decimal.
c) Calculate the average value of the above N numbers.
d) Print N and the result – the printing format should be aesthetic.

Guidelines:

a) Data: the program should have the following variables:
   - N is positive: you can use long long.
   - A variable to store a real-valued number in decimal: you can use float.
   - A variable to store the average value: it can be double.
b) Algorithm:

- Use a while loop to check if N is positive. Prompt the user until he/she enters the correct number.
- Use `<srand>` library and `<time>` to initialize the random function.
- Use `rand` to generate random numbers, combined them with a for loop to calculate the average value.

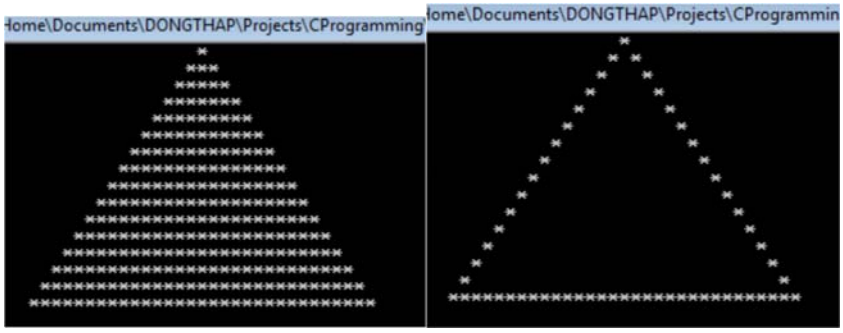
Exercise 6. Write a program that allows the user to:

a) Enter a positive integer N, only stops when the user enters the correct number.
b) Calculate the factorial N!.
c) Print N and the result – the printing format should be aesthetic.

Guidelines:

a) Data: the program should have the following variables:
   - N is positive: you can use long long.
   - A variable to store the factorial result: unsigned long long.
b) Algorithm:
   - Use a while loop to check if N is positive. Prompt the user until he/she enters the correct number.
   - Use a for loop to calculate N!.
   - Print the result.

Exercise 7. Print two isosceles triangles, a filled one and a hollow one. The user needs to enter the length of the base of the triangle (positive odd integer).



Guidelines:

a) Data: We need a variable N to store the length of the base and another variable H to store the height of the triangle.
b) Algorithm:
   - Notice that the number of lines we will print is also the height of the triangle. From the base length (N), we can deduce how many lines we need

to print (H). From this, we can use a loop to print the triangle line-by-line. Because we already how many lines we are going to print, a for loop is appropriate.

- Looking at the filled triangle in the example figure, the middle '*' in the first line has a spacing of N/2-1 from the left, the first '*' of the second line has a spacing of N/2-2 from the left, and so on. For each line, from the top of the triangle, the number of '*' will gradually increase from 1, 3, 5 to N. With these two clues, you can print the entire filled triangle using two nested loops.
- To print the hollow triangle, we can print the first line and the last line just as before. These two lines are different therefore we can print them outside of the loop. For the rest, we only need to print the first '*' and the last '*' in each line.

Exercise 8. Write a program to print the n^{th} Fibonacci number, given that:

$$F(0) = 1$$

$$F(1) = 1$$

$$F(n) = F(n-1) + F(n-2), n \geq 2$$

Guidelines:

a) Data: the program should have the following variables:
   - n: float (to verify the requirement that n must be a non-negative integer).
   - previous, current, next: int.

b) Algorithm:
   - Check the value of n. The requirement is that n must be a non-negative integer. When user inputs invalid values (character, negative number, etc.) the program must print an error message.
   - Initialize previous, current and next with appropriate values.
   - Iterate from 0 to n and update previous, current and next to appropriate values.
   - Print the value of F(n), be wary of the case where n = 0 and n = 1.

Exercise 9. Write a program to calculate sin(x). The program expects the user to enter the angle x in degree.

Guidelines:

The function sin(x) can be approximated using Taylor series:

$$\sin x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1} = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \cdots$$

a) Data: x is a real number hence we can choose its type to be either float or double.
b) Algorithm:
- Decide the value for epsilon (a very small number), for instance 10^{-10}.
- Notice that on the right-hand side of the equation, the expression can be split into n smaller terms.
- We can calculate sin(x) by accumulating the operands using loop, each loop we will calculate the corresponding term and add it to the total result. The loop will stop when the current term is small enough (compared to epsilon). The smaller epsilon is, the more accurate our result will be,
- For each term, we need to calculate these three:
  - $A = x^{2n+1}$
  - $B = (2n+1)!$
  - $C = (-1)^n$
- The term will be C * A/B.
- Sum of all terms from each loop will be the sin(x) we want to know.
- **Note:** the number x in the equation is in radian but we let the user enters in degree therefore we need to convert x from degree to radian before computing sin(x).

Exercise 10. Write a program to calculate cos(x), tan(x) and ln(x).

Guideline:

The functions cos(x) and ln(x) can be approximated as follow:

$$\cos x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!} x^{2n} = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \cdots$$

$$\ln x = 2\left[\left(\frac{x-1}{x+1}\right) + \frac{1}{3}\left(\frac{x-1}{x+1}\right)^3 + \frac{1}{5}\left(\frac{x-1}{x+1}\right)^5 + \cdots\right] \qquad x > 0$$

a) Data: x is a real number therefore its type can be float or double.
b) Algorithm: you should refer to Exercise 9.

Exercise 11. Write a program to calculate a polynomial of order N.

Guideline:

A polynomial of order N can be written as follow: $A_0 + A_1X + A_2X^2 + A_3X^3 + ...$

a) Data: The use should be allowed to enter:
   - N: choose an appropriate data type for non-negative integer.
   - X: real valued.
   - N real valued coefficients: real valued.

b) Algorithm:
   - Consider the following polynomial: $A_nX^n + A_{n-1}X^{n-1} + A_{n-2}X^{n-2} + ... + A_1X^1 + A_0$.
   - The user should enter all of the coefficients in a single string "A_n A_{n-1} A_{n-2} ... A_0" (Note, there must be a space between two numbers). You should use stringstream to read all of them.
   - After the coefficients, the user can enter x. The program should be able to check if x is a valid real number (not character). If it's not, prompt the user to enter x until it is valid.

To calculate the polynomial:

- Represent it as follow:

$[[A_n * X + A_{n-1}] * X + a_{n-2}] * X + ...$

- With the above clue, we can accumulate the polynomial result like this:

$P_x = P_x * x + term$

- Initialize $P_x = 0 \rightarrow$ we can then calculate the polynomial as follow:
  - At first $P_x = A_n * X + A_{n-1}$
  - Then, $P_x * x + term = P_x * x + A_{n-2} = [[A_n * X + A_{n-1}] * X = A_n * X^2 + A_{n-1} * X$.
  - And so on, until we reach A_0.

A detailed example with $3x^2 + 4x + 5$:

- The user inputs 3 4 5 as coefficients.
- The user inputs 2 as X.
- Initialize $P_x = 0$.

The loops:

- Loop 1:

- Read the coefficient // coeff = 3
- $P_x = P_x * X + coeff$ // $P_x = 0 * 2 + 3 = 3$

- Loop 2:
  - Read the coefficient // coeff = 4
  - $P_x = P_x * X + coeff$ // $P_x = 3*2 + 4 = 10$
- Loop 3:
  - Read the coefficient // coeff = 5
  - $P_x = P_x * X + coeff$ // $P_x = 10*2 + 5 = 25$
- The result is 25.

Exercise 12. Write a program to:

a) Print a menu with the following options:

```
1. Enter product
2. Find product
3. Print the list of products
4. Remove product
5. Update product
6. Save data
7. Load data
8. Quit

Please choose an option:
```

b) Read the option entered by the user. If the user entered an incorrect choice, tje screen must be cleared and re-display again.

Guidelines:

a) Data: you should refer to Exercise 6 in lab 3.
b) Algorithm:

The program keeps re-printing the menu until user enters a correct input. In this case, a loop is appropriate. Since the program prints the menu beforehand and allows the user to enter the choice before checking, a do-while loop is the most suitable. The printing is identical to Exercise 6 in lab 3. However, if the user entered an incorrect value our program will have to re-print the menu. You can consider using an additional boolean variable as the stopping condition of the loop.

To clear the console screen, you can use system("cls").