C++ Lab5 (2 classes)

2D vectors – Friends and Facebook

Objectives

1. Create and fill two-dimensional vectors.
2. Loop over and read the contents of 2D vectors.
3. Use Booleans.
4. Handle different types of loops.

General comments

1. Carefully read each question before writing code.
2. Prepare a set of test cases for each question before writing code.
3. When writing programs, look at what already exists.
4. Test using your set of test cases. These should be added as comments at the end of your code.

WORK TO HAND IN: upload your program to Moodle, including test cases.

In this lab we will simulate a Facebook network with a 2D vector named **friends**. Lines and columns represent the members registered on the network and each element [**i**][**j**] will contain a Boolean indicating if **i** and **j** are friends (true if that is the case, false otherwise). Members of the network will be identified by numbers starting at 0. (This is obviously far from reality…)

Example: with 4 members

true	false	true	true
false	true	true	false
true	true	true	true
true	false	true	true

Sample display:

```
Member 0 is friends with: 0 2 3
Member 1 is friends with: 1 2
Member 2 is friends with: 0 1 2 3
Member 3 is friends with: 0 2 3
```

Question 1

Create a new project using the program **faceBook-Lab5.cpp** available on Moodle and add statements to initialize the **friends** network with 4 members in such a way that **each member is friends with themselves (and no one else).**

Question 2

Display the contents of the vector in the form given above (one line per person, containing: "Member X is friends with: X Y Z").

Question 3 (optional)

Modify your program to display only the friends other than themselves. Careful: use 2 successive loops rather than a single one with a test on the index …

Question 4

Extend the program to update the **friends** vector in order to memorize that 2 members are now friends. The identification number of these members is given by the user. Check the validity of these numbers with an input loop. How many vector elements need to be updated? Check your code by displaying the contents of the vector. Add several new friendships and check the result. Use a loop: "Do you want to continue? y/n")

Question 5

Compute and display the number of the member who has the most friends (in the above example it is person 2). In case of equality between several members display the first.

Question 6 (optional)

Extend the program to display the number of friends that are common between two members. The identification number of these members is again given by the user (check their validity with an input loop). In the above example, the number of common friends between members 1 and 3 is 3 (because the friends that are common to 1 and 3 are 1, 2 and 3).

Question 7 (optional)

The **friends** vector is correct (i.e. coherent) only if it is symmetric (if **i** is friends with **j**, then **j** is friends with **i**). Extend your program to check and display if the vector is correct.

Question 8 (optional)

We want to be able to identify members by their names rather than numbers. Add a **names** vector to your program containing the name of all the members (same size as the **friends** vector). The correspondence between names and numbers will be done through vector indices (the number of a member corresponds to their index in the **names** vector). Then modify the display statements of question 1 (and the following ones if you have the time) to display members' names rather than their numbers.