3 Debugging

Objective: to practice debugging a C++ program

1 A (broken) system for access control

A company has created a system for controlling access to its facilities. To open a door, an employee inserts a card in a card-reader. The system looks up the card number in a table. If the card number is found, the name of the employee is presented on a display and the door is unlocked. Otherwise the door remains locked.

The class `UserTable` is used to check who a certain card number belongs to. The access control system includes the following code:

```
UserTable t;
...
int cardNbr = sensor.getCardNbr();

User u = t.find(cardNbr);          // Find the owner of the card

if (u == UserTable::user_not_found) {
  display.showText("** access denied **");
} else {
  display.showText("Welcome, " + u.getName());
  door.unlock();
}
```

The interesting line is marked with a comment. Given a card number (an integer) the system looks up a user. The variables `sensor`, `display`, and `door` refer to objects outside the scope of this task. Can you understand the general operation of the above code?

The class `UserTable` is unfortunately broken. Your task is to find and correct the mistakes. Present your solution according to the table at the end of this section, together with a working test program.

2 Tasks

Study the source code in the files *User.h*, *User.cpp*, *UserTable.h* och *UserTable.cpp*. There is also a text file, *users.txt*, containing all users and their card numbers. Look at the file and see how it is organized.

In the class `UserTable` the user database is read from *users.txt* and stored in an array.

We will now debug the class `UserTable`**.** To test the class, it is impractical to try a lot of cards in the card reader. A better option is to create a simple test-program that creates a `UserTable` object and calls the function `find` in the same way the card-reader does. Then you can simply make up and test different card numbers yourself.

Create such a test program (including a `main` function) that

- creates a `UserTable` object,
- calls the function `find(int)` to find the user with card number 21330,
- calls the function `find(int)` to find the user with card number 21331 and
- calls the function `find(std::string)` to find the user named ”Jens Holmgren”.

What is the result of the program? What did you expect? (Compare with the file *users.txt*.)

The class `UserTable` has a member function for printing the contents of the table. Extend your test program with a call to this function and see what is printed.

Study the constructor for the class `UserTable`. Here, the user database is read from a file and stored in an array. Either the reading from the file is not working, or the read lines are not stored correctly in the array.

Add some lines to your test program so that a new (made up) user with card number 1234 is added, and then print the contents of the table. Run your test program. How many users are now in the table? How many did you expect?

Find and fix the mistake that you identified in the previous task. To get further clues you can ask yourself the following questions:

- The function `add` inserts a `User` object at the right position in the array. What positions are the objects put in? That is, what value does the variable `pos` get? (You can either use debug printouts, or run it in the debugger and use breakpoints.)
- Does the function `getNbrUsers` return the correct result? (Test.)

When you have corrected the mistake, check if searching for a card number works. Verify that your test program calls `find(int)` with a card number, and that the result is the same card number and the correct user name. Compare with the corresponding line in *users.txt*. Always compile with warnings enabled (the option `-Wall` to `g++`).

It has been reported that `find(std::string name)` does not work. The function is implemented with *linear search*: starting at the first position, go through the array until a person with the name `name` is found. Study the implementation of the function to find the bug.

When the function is corrected, the search for Jens Holmgren should work.

Verify that searching works. There is function `testFindNumber()` that can be used to test the member function `UserTable::find(int)`. Let your test program call that test function and verify that all users (card numbers) now can be found.

Check the test function. It has been reported that `testFindNumber()` performs the tests correctly, but that the tested `UserTable` object is left in an unusable state afterwards. Investigate what happens when `testFindNumber()` is called.

Note. The names in the table have been generated from the 100 most common swedish family names and male and female names[8] (excluding names containing `åäö`). They do not refer to actual persons.

[8] Source: SCB, `http://www.scb.se/namnstatistik/`.

Solution

To pass the task you have to show a working program, according to the above instructions. In addition you must be prepared to account for your debugging work. Record information about each bug so that you remember them when you present your solution.

Bug (what function?, what line?)	*Describe the problem.*	*How did you find the bug?*