

## Lab 3.10.1 Using pseudorandom values - a little lottery

### Objectives

Improve the student's skills in:

- using standard library functions;
- using vectors;
- and familiarize the student with the concept of pseudorandom number generation and use.

### Scenario

Don't think we want to encourage you to start gambling – nothing could be more wrong. We only want you to write a code that tries to "predict" (note the quotes) the numbers for the lottery.

There are many different lotteries so your program should be flexible. It will have to know two basic parameters: how many balls are in the machine and how many of them are drawn. Therefore, your code must input two int values reflecting these restrictions.

Next, the program should "draw" (note the quotes again) the required number of balls. To simulate the process of drawing, we'll use the so-called "pseudo-random number generator" – an algorithm producing a series of numbers which behave as if they were "drawn", although the values are strictly deterministic and (sad but true) completely predictable.

Here's a short piece of code showing you how to use it:

```
#include <iostream>
#include <cstdlib>
#include <ctime>

using namespace std;

int main(void) {
    int max = 10;

    srand(time(NULL));
    int rnd = rand() % max + 1;
    cout << rnd << endl;
    return 0;
}
```

- the "cstdlib" header file is needed as it provides two functions: srand() and rand();
- the "ctime" header file is needed too, because we are going make use of the time() function;
- the "srand()" function initializes the generator – passing the current clock value to the function makes behavior of the generator more random, as the clock value is rather unpredictable;
- invoking the "rand()" function gives us a pseudo-random number from the range [0 - RAND\_MAX] (note the brackets: they says that 0 and RAND\_MAX are included in the range too); the actual value of the RAND\_MAX symbol is implementation dependent and we don't need to know it, as we want to "normalize" the returned value to our own range: [1..max]; analyze our code carefully and guess how we made it;
- compile the code and run it several times – you'll see it outputs a number that seems to be "random" (in other words: you won't be able to predict it).

And here goes the code you should complete. We want it to behave in the following way:

- it should input two int values (you get this part free of charge);
- next, it should "draw" as many balls as the user wants, but remember – you can't use the same number more than once! This means that you have to create an array for all the numbers that have already been drawn (use the "new" operator!) and check whether the new number hasn't been used already;
- finally, you should output your numbers and free up the previously allocated memory (using the delete[] operator)

Test your code carefully. This isn't gambling!

```
#include <iostream>
#include <cstdlib>
#include <ctime>

using namespace std;

int main(void) {
    int maxball;
    int ballsno;

    cout << "Max ball number? ";
    cin >> maxball;
    cout << "How many balls? ";
    cin >> ballsno;
    srand(time(NULL));

    // Insert your code here

    return 0;
}
```