Practical Work nr. 3 – Cycles

Subjects

- Repetition statements (loops)

Exercices

1. Write a program that asks you for real numbers until you enter the number zero. At the end, indicate the maximum value, the minimum value, the average and the number of elements entered (not counting the stop value).

2. The HighLow game consists of trying to guess an integer higher than or equal to 1 and smaller than or equal to 100. The program picks a number at random. Then the user enters an attempt and the program indicates if it is too high or too low. This is repeated until the user hits the number. The game ends up indicating how many attempts have been made. For this problem use the following code as a starting point (this code already calculates the secret number). You can choose the option C in case the option B always generate the same value, since it depends on the properties of your computer and Operating System).

```
#include <chrono>
#include <iostream>
#include <random>
using namespace std;
int main(){

    // Option A
    srand(time(NULL));
    cout << 1 + rand() % 100 << endl;

    // Option B
    std::random_device rd;
    std::uniform_int_distribution<int> dist(1, 100);
    cout << dist(rd);

    // Option C
    auto seed = chrono::system_clock::now().time_since_epoch().count();
    seed_seq sseq{seed};
    mt19937 generator(sseq);
    uniform_int_distribution<int> dist(1, 100);
    unsigned secret = dist(generator);
    cout << secret;

    return 0;
```

}

3. Write a program that reads a positive integer from the keyboard and determines if the number entered is a prime number. A natural number is a prime number when it has exactly two distinct factors: one and itself. Tip: Try dividing the number by 2, 3, etc. If it is divisible by any number smaller than itself, then it is not prime. Pro tip: if n = p * q then at least one of p or q cannot be larger than the square root of n; so, it is enough to try integer divisors up to the square root of n.

4. Write a program that reads from the keyboard a positive integer, N, and prints out the list of all its own proper divisors (all integers that divide N except N itself). The program should also indicate whether N is a deficient, perfect, or abundant number. Keep in mind that:
    a. a number is said to be **deficient** if the sum of its proper divisors is smaller than the number itself (example: 16 --- 1+2+4+8<16);
    b. a number is said to be **perfect** if the sum of its proper divisors is equal to the number itself (example: 6 --- 1+2+3=6);
    c. a number is said to be **abundant** if the sum of its proper divisors is larger than the number itself (example: 18 --- 1+2+3+6+9>18).

5. **Challenge.** Write a program that attempts to guess a secret bi-dimensional position, selected by the user of the program. Each coordinate is an integer that belongs to the interval [1,100]. After each guess made by the program, the user must indicate the relative position of the guess with respect to the secret coordinate using an integer code, as illustrated in the following table:

0 (NW	1 (N)	2 (NE)
3 (W)	4 (Correct)	5 (E)
6 (SW)	7 (S)	8 (SE)

For example, an answer of 8 (SE) means that the x coordinate of the guess is too large (to the East of the secret) and that the y coordinate of the guess is too small (to the South of the secret).