Lesson 4

Arrays

Arrays

Variable is like a cell in a storage room

But what if we want to take an arbitrary amount of these cells?

An array is special type of a variable. It contains several variables of the same type at once.

Moreover, the elements of the array are ordered - each has its own number.

Only the entire array has a name, the elements have only an ordinal number in this array

Arrays

Items are numbered starting at zero. The number of an element in an array is called its index

a =

0	1	2	3	4	5
124	5326	37345	34	15	-12

Array size is 6!

You can think of the index as the distance from the leftmost element. On the ruler, lines are also numbered from zero.

An array element is accessed through square brackets: a [i]

```
std::cout << a[1] << std::endl;
a[2] = 0;
a[3] = a[2] + a[1];
std::cout << a[3] << std::endl;
```

Arrays

When creating an array, you need to specify its size:

```
int a[6];
```

By default, as in the case of an uninitialized variable, all array elements contain so-called garbage. That is, each element has no specific meaning. Therefore, the arrays need to be initialized:

```
a[0] = 1;
a[1] = 2;
a[2] = 3;
a[3] = 4;
a[4] = 5;
a[5] = 6;
```

Array and for loop are best friends =)

Let's try to *initialize* the array with numbers from 1 to 10:

```
int a[10];
for (int i = 0; i < 10; i ++)
    a[i] = i + 1
```

Practice

Task 1: A sequence of n numbers is entered on the keyboard. Print numbers in reverse order.
Input: First, the number n itself, then a sequence of n numbers
Output: The same n numbers in reverse order

Task 2: A sequence of n numbers is entered on the keyboard. Determine if it is a palindrome.
Input: First, the number n itself, then a sequence of n numbers
Output: Whether the string is a palindrome or not

Task 3: A number n is given, which is the size of a square matrix. It is necessary to assign to each diagonal its distance from the main one.
Input: the number n
Output: the required matrix

Arrays and const qualifiers

If we combine array modifier and const qualifier, we'll create a const-qualified array.

```
const int a[6];
```

But in this case, we'll get a compilation error, because a is uninitialized.

In order to initialize it, we should use initializer lists:

```
const int a[6]{1, 2, 3, 4, 5, 6};
```

or:

```
const int a[6] = {1, 2, 3, 4, 5, 6};
```

Declaration point of an array

```
int x[x];
```

Compilation error, x is undefined.

```
int x = 2;
{
    int x[x];
}
```

Ok, array size is 2

Arrays

I would like to be able to use arrays even if the size is large.

In the case of a regular array, if you use a large size, a RunTime Error (Segmentation Fault) will occur.

```
int a[150]; /// OK
int a[10000000]; /// RE
```

So how to create large arrays?

Also:
a [150] = 3;
This code will work, but why ???

Moreover, even the following code works: a[-5] = 4;
Let's figure out what happens at the physical level.