Unit 5: Arrays

1. Introduction

Question: Suppose user enters odd number of numbers from keyboard, and program stops when she enters 0. The program writes the number she entered in the middle. How can you write the program?

Sample input:

```
3 9 123 11 12 99 0
```

Sample output:

```
11
```

In the example above user enters 7 numbers (including 0). The number in the middle is 11 since it is the 4th number.

An array is a sequence of boxes. An array has a name and size like a variable. The size of the array is the number of boxes in the array. The boxes are numbered from 0 to n – 1 where n is the size of the array.

Example.

myArray

0	1	2	3	4	5
3	7	12	4	8	11

An array is shown above. The name of the array is *myArray* and its size is 6. 0th value of the array is 3, and 4th value is 8.

2. Declaring Arrays

We declare arrays similar to variables. The only difference is that the size is indicated in square brackets.

```
int myArray[20];
```

The above code declares an array of size 20, and it is named `myArray`. The entries (boxes) of the array are numbered from 0 to 19. All the entries of an array are the same type. For this example, each entry of the array is integer. Similarly, we can declare an array of decimal values as follows:

```
float myArray[20];
```

Array initialization is a little different. For instance:

```
int numbers[3] = {3, 5, 123};
```

puts 3, 5, and 123 in the array in order.

3. Writing and Reading Arrays

We can use arrays similar to the variables. Suppose we have an array named myArray with the size 10. In order to put 7 into 5th location of the array, we write the following code:

```
myArray[5] = 7;
```

The only difference from variables is we always indicate the location in the arrays. In this example location is indicated in square brackets.

We can use the number in a location of the array similar way. Suppose we have a variable named `apple` and an array named `banana`. To assign the value 6 more than the 4th value of `banana`, we write the following code:

```
apple = banana[4] + 6;
```

We can use all arithmetic operations we learned before on the values of an array.

We can also use a variable to indicate the location of an array. For example, suppose the variable x equals to 4:

```
myArray[x] = 3;
```

This code puts 3 into 4th location of `myArray`.

Example. The program below defines an array of 6 locations and reads them from the keyboard. Then it adds the 0th and 3rd values, and puts into 5th location in the array. After that, it writes the 5th value to the screen.

Sample input:

```
3 9 123 11 12 99
```

Sample output:

```
14
```

```
#include <iostream>
using namespace std;

int main()
{
  int myArray[6];
  cin >> myArray[0];
  cin >> myArray[1];
  cin >> myArray[2];
  cin >> myArray[3];
  cin >> myArray[4];
  cin >> myArray[5];

  myArray[5] = myArray[0] + myArray[3];
  cout << myArray[5] << endl;
}
```

Question: If we need to read 100 numbers from the keyboard and put them into the array, should we do it one by one?

Exercise 1: Write a program that reads numbers from the keyboard, and outputs the number in the middle. The program stops when the user enters 0. It is guaranteed that the user inputs odd number of numbers, and there are at most 21 numbers (including 0).

Sample input:

```
3 9 123 11 12 99 0
```

Sample output:

```
11
```

Exercise 2: Write a program that reads n numbers from the keyboard where $n \leq 20$. Then it writes these numbers to the screen in reverse order. The user will enter n from the keyboard.

Sample input:

```
6
3 9 123 11 12 99
```

Sample output:

```
99
12
11
123
9
3
```