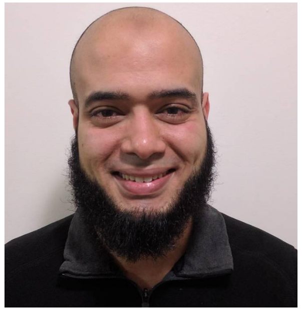
Programming4**kids**

1D Arrays

Mostafa Saad Ibrahim
Computer Vision Researcher @ Huawei Canada
PhD - Simon Fraser University
Bachelor / Msc - FCI Cairo University

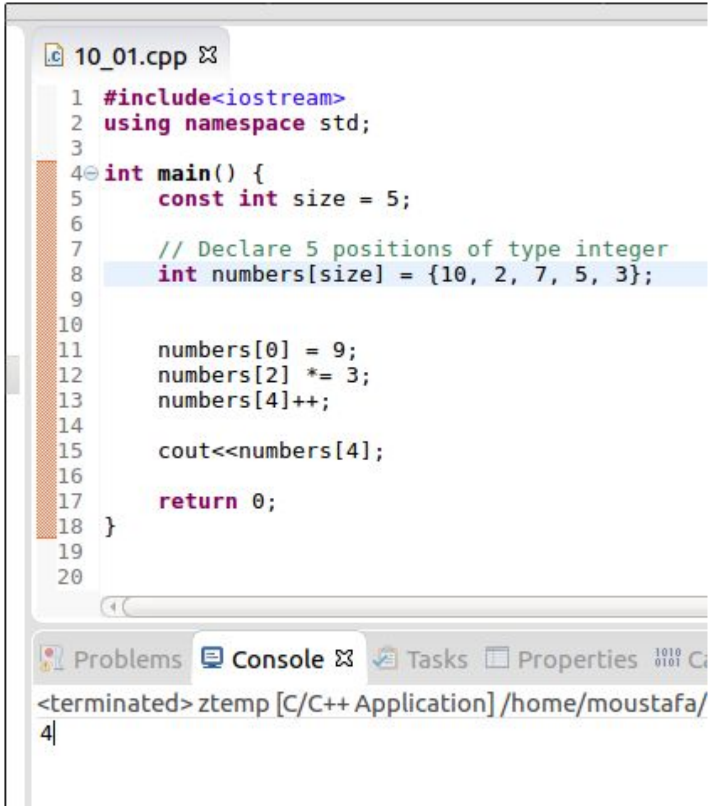
Ex-(Software Engineer / Teaching Assistant)



Write a program that:

- That reads 1000 integers and print them reversed!
- That reads 1000 integers and find pairs of numbers with sum 12345?
- We can define 1000 variables! But this is a crazy idea!
- Programming languages introduce datatype **array** of **size K**
  - K variables defined in the memory (consecutively)
  - They all of **same data** type
- So now we create an array of size 1000
  - Then print them reversed!
  - That is all

Declare an array



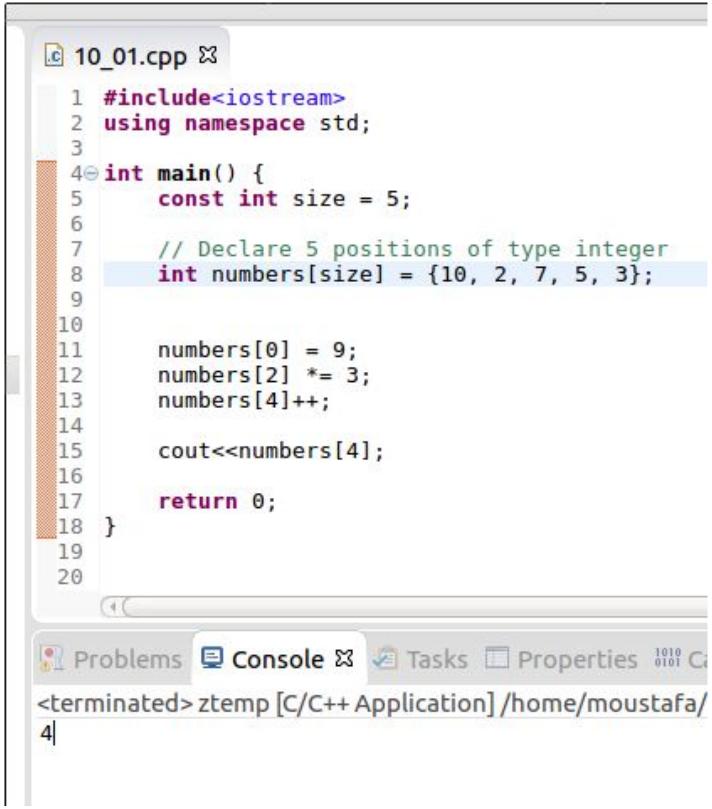
10_01.cpp

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

int main() {
    const int size = 5;

    // Declare 5 positions of type integer
    int numbers[size] = {10, 2, 7, 5, 3};


    numbers[0] = 9;
    numbers[2] *= 3;
    numbers[4]++;

    cout<<numbers[4];

    return 0;
}
```

```
<terminated> ztemp [C/C++ Application] /home/moustafa/
4
```

- Int number = 10;
- Int numbers**[5]**;
  - Create 5 numbers (variables)
  - Type integer
- numbers[i]
  - Access **ith** number
  - Completely like normal variable
  - We can read/output/change
- **Zero indexing**
  - numbers[0] first variable
  - numbers[size-1] last variable

Declare an array



10_01.cpp

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

int main() {
    const int size = 5;

    // Declare 5 positions of type integer
    int numbers[size] = {10, 2, 7, 5, 3};


    numbers[0] = 9;
    numbers[2] *= 3;
    numbers[4]++;

    cout<<numbers[4];

    return 0;
}
```

```
<terminated> ztemp [C/C++ Application] /home/moustafa/
4
```

- Line 8 declare the array

Index	0	1	2	3	4
numbers	10	2	7	5	3

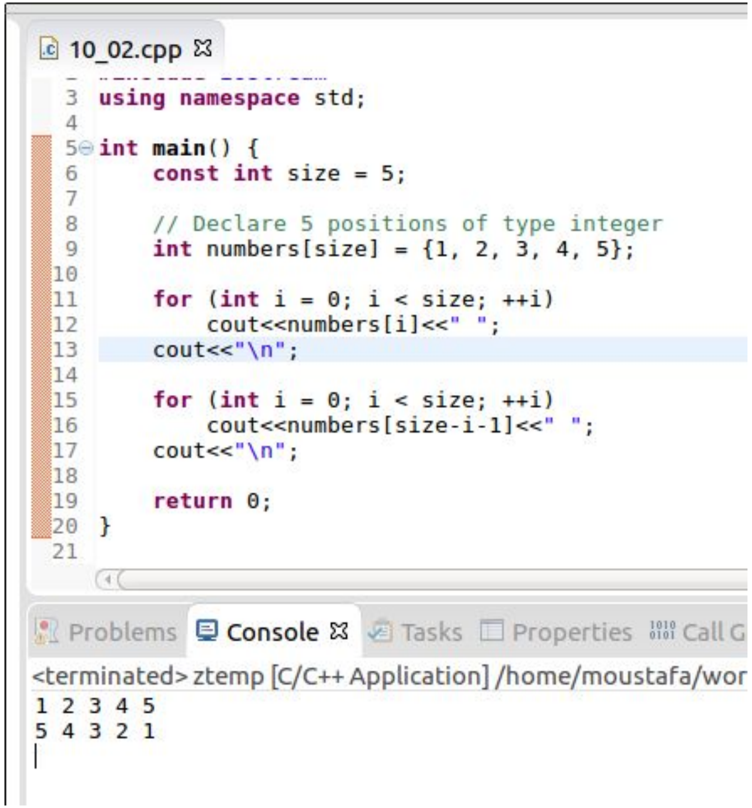
- Line 11 changes first number to 9

Index	**0**	1	2	3	4
numbers	**9**	2	7	5	3

- Line 12 and 13 also do changes

Index	**0**	1	**2**	3	**4**
numbers	**9**	2	**21**	5	**4**

Printing array forward and backward



10_02.cpp

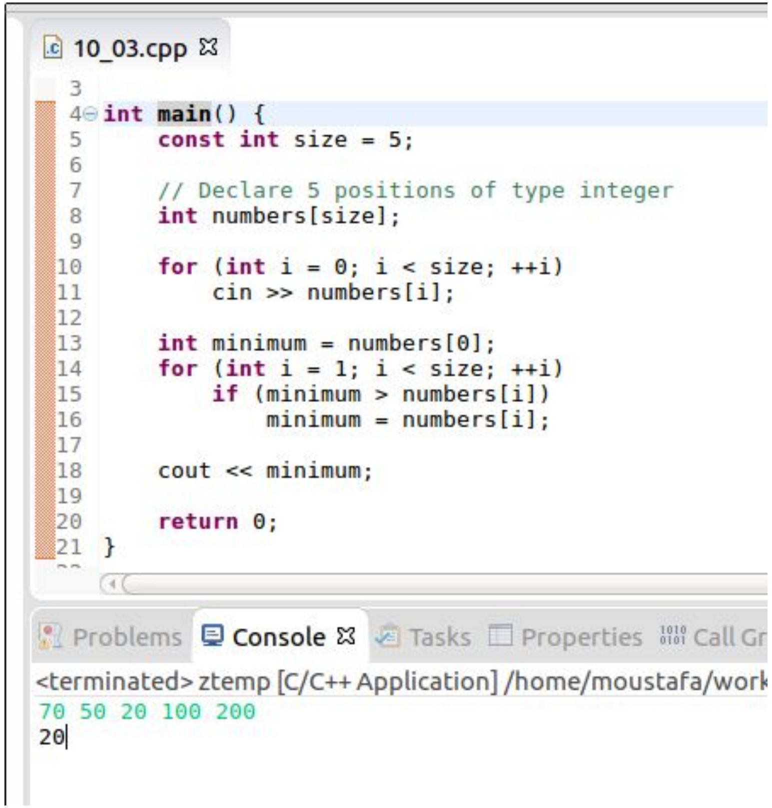
```cpp
using namespace std;

int main() {
    const int size = 5;

    // Declare 5 positions of type integer
    int numbers[size] = {1, 2, 3, 4, 5};

    for (int i = 0; i < size; ++i)
        cout<<numbers[i]<<" ";
    cout<<"\n";

    for (int i = 0; i < size; ++i)
        cout<<numbers[size-i-1]<<" ";
    cout<<"\n";

    return 0;
}
```

Problems Console Tasks Properties Call G

```
<terminated> ztemp [C/C++ Application] /home/moustafa/wor
1 2 3 4 5
5 4 3 2 1
```

- Remember last element position is size-1
- Trace the backward
    - Index 4
    - Index 3
    - Index 2
    - Index 1
    - Index 0

Read 5 numbers in array - find minimum



```cpp
int main() {
    const int size = 5;

    // Declare 5 positions of type integer
    int numbers[size];

    for (int i = 0; i < size; ++i)
        cin >> numbers[i];

    int minimum = numbers[0];
    for (int i = 1; i < size; ++i)
        if (minimum > numbers[i])
            minimum = numbers[i];

    cout << minimum;

    return 0;
}
```

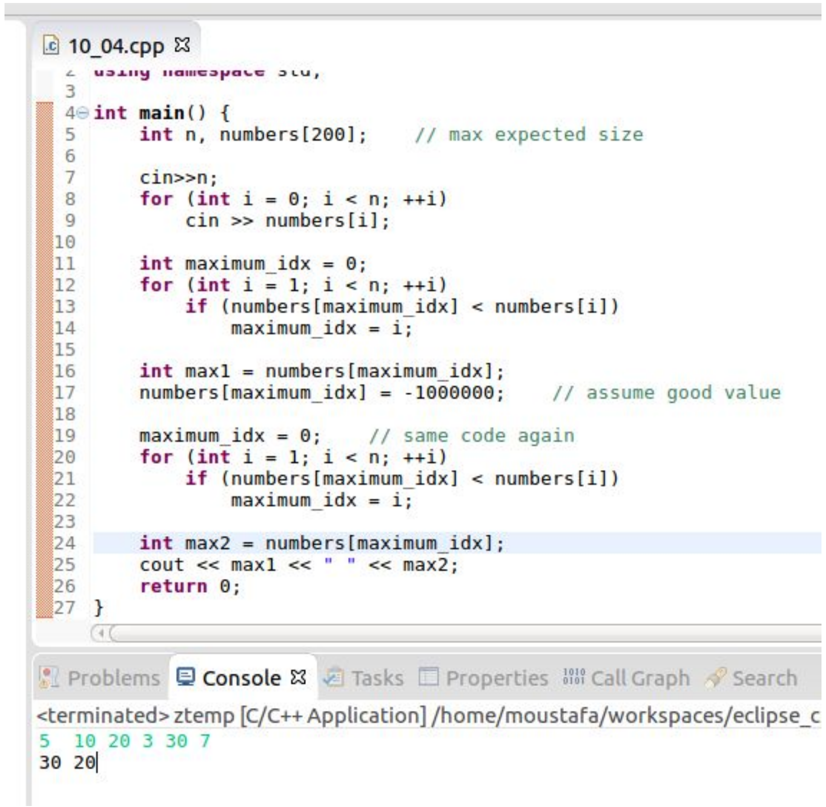
```
<terminated> ztemp [C/C++ Application] /home/moustafa/work
70 50 20 100 200
20
```

- Remember: Deal with each cell as a variable
    - read/write/assign

Practice: Find first and 2nd Maximum values

- Read an Integer N (< 200), then read N (distinct) integers. Find the maximum and 2nd maximum values
- Input: 5 10 20 3 30 7   ⇒ Output 30 20
    - 30 is the maximum in the array
    - If we removed it, the next maximum is 20
- Stop the video and code it

Practice: Find first and 2nd Maximum values



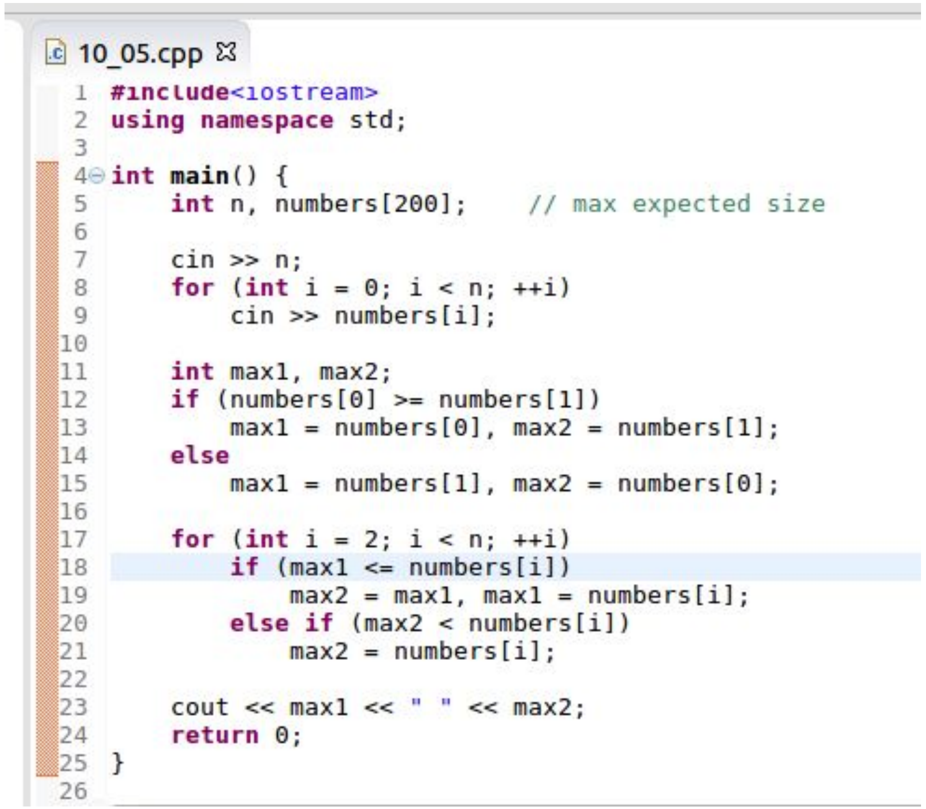
10_04.cpp

```
using namespace std;

int main() {
    int n, numbers[200];    // max expected size

    cin>>n;
    for (int i = 0; i < n; ++i)
        cin >> numbers[i];

    int maximum_idx = 0;
    for (int i = 1; i < n; ++i)
        if (numbers[maximum_idx] < numbers[i])
            maximum_idx = i;

    int max1 = numbers[maximum_idx];
    numbers[maximum_idx] = -1000000;    // assume good value

    maximum_idx = 0;    // same code again
    for (int i = 1; i < n; ++i)
        if (numbers[maximum_idx] < numbers[i])
            maximum_idx = i;

    int max2 = numbers[maximum_idx];
    cout << max1 << " " << max2;
    return 0;
}
```

```
<terminated> ztemp [C/C++ Application] /home/moustafa/workspaces/eclipse_c
5  10 20 3 30 7
30 20
```

- Easy idea
- Find first maximum (idx)
  - Mark with very small value
- Find again first maximum
  - This is now the 2nd maximum
- Disadvantages
  - Need to loop twice
  - Need good value for -ve
    - Workaround: ignore previous position

Practice: Find first and 2nd Maximum values



10_05.cpp

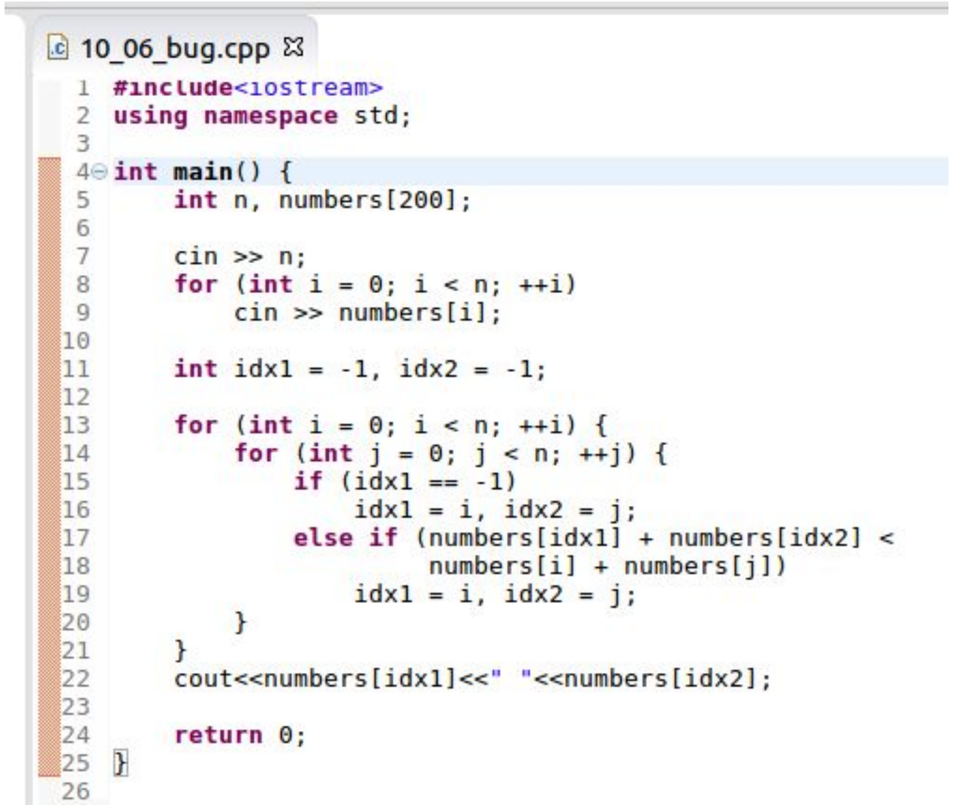
```cpp
#include<iostream>
using namespace std;

int main() {
    int n, numbers[200];    // max expected size

    cin >> n;
    for (int i = 0; i < n; ++i)
        cin >> numbers[i];

    int max1, max2;
    if (numbers[0] >= numbers[1])
        max1 = numbers[0], max2 = numbers[1];
    else
        max1 = numbers[1], max2 = numbers[0];

    for (int i = 2; i < n; ++i)
        if (max1 <= numbers[i])
            max2 = max1, max1 = numbers[i];
        else if (max2 < numbers[i])
            max2 = numbers[i];

    cout << max1 << " " << max2;
    return 0;
}
```

- Maintain 2 variables for the 2 maximums
- Iterate on the array and update together
- Say we have so far 20 10
  - And we found value 30
  - Now we should be 30 20
- Say we have so far 20 10
  - And we found value 15
  - Now we should be 20 15

Practice: Find pair values of maximum sum

- Read an Integer N, then read N <= 200 (distinct) integers. Find a pair of numbers (e.g. 2 different indices) whose sum is maximum
- Input: 5 2 10 3 50 15 ⇒ 65 (from 50 + 15)
- Stop the video and code it

Practice: Find pair of max sum - Buggy

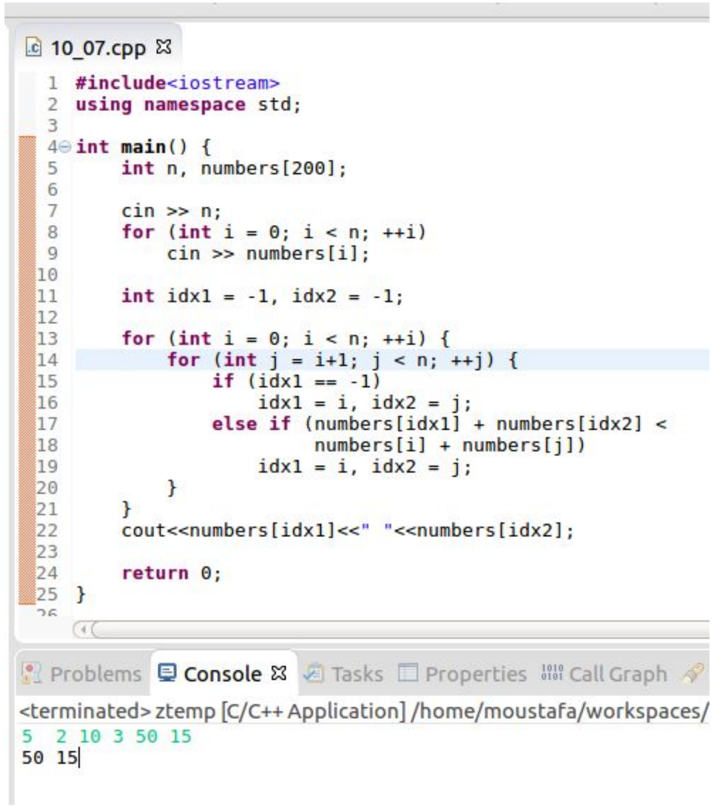


10_06_bug.cpp

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

int main() {
    int n, numbers[200];

    cin >> n;
    for (int i = 0; i < n; ++i)
        cin >> numbers[i];

    int idx1 = -1, idx2 = -1;

    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            if (idx1 == -1)
                idx1 = i, idx2 = j;
            else if (numbers[idx1] + numbers[idx2] <
                     numbers[i] + numbers[j])
                idx1 = i, idx2 = j;
        }
    }
    cout<<numbers[idx1]<<" "<<numbers[idx2];

    return 0;
}
```

- Let’s just do 2 nested loops to find the pair
    - There is a bug
    - Also half of operations is useless!

Practice: Find pair of max sum - Fixed



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

int main() {
    int n, numbers[200];

    cin >> n;
    for (int i = 0; i < n; ++i)
        cin >> numbers[i];

    int idx1 = -1, idx2 = -1;

    for (int i = 0; i < n; ++i) {
        for (int j = i+1; j < n; ++j) {
            if (idx1 == -1)
                idx1 = i, idx2 = j;
            else if (numbers[idx1] + numbers[idx2] <
                    numbers[i] + numbers[j])
                idx1 = i, idx2 = j;
        }
    }
    cout<<numbers[idx1]<<" "<<numbers[idx2];

    return 0;
}
```

```
<terminated> ztemp [C/C++ Application] /home/moustafa/workspaces/
5  2 10 3 50 15
50 15
```

- Trick: Start j from i+1
  - Avoid duplicate bug
  - Avoid duplicate processing
    - We test positions (2, 4) and then test (4, 2) which is same locations!
- This is very inefficient code!
  - Can you do it using a single loop!
    - Hint: Simple observation

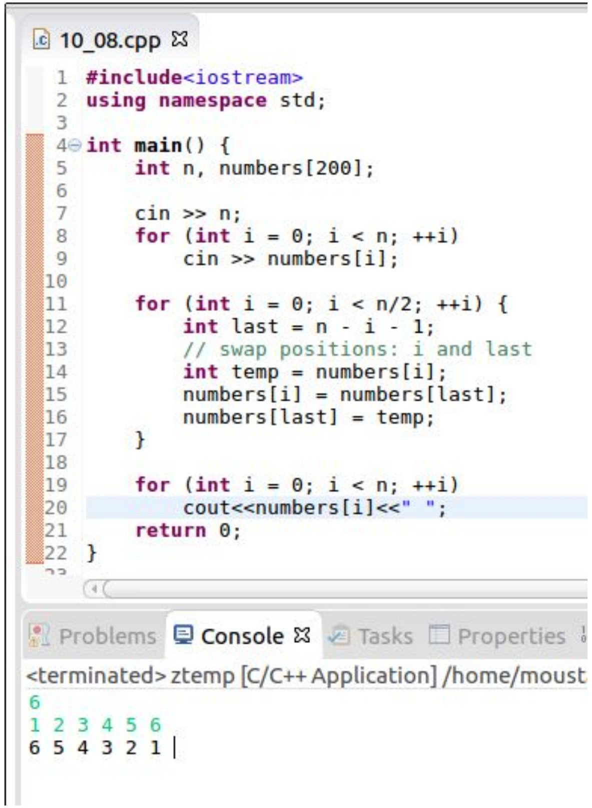
Practice: Find pair of max sum - FASTER

- Simply, the pair of maximum sum must come from **the maximum value and the 2nd maximum value**
- Use the code we explained, get them and sum them
- Think more ⇒ Code efficient

Practice: Reverse in place

- Read an Integer N, then read N <= 200 integers.
  - In-place: Change the current array, don’t use 2 arrays
- Simple idea: Iterate from the begin and end in same time
  - Swap the 2 positions
  - Do this tell the middle only
- Let say array is 1 2 3 4 5 6 7 8
  - Step 1: swap (1, 8) ⇒ 8 2 3 4 5 6 7 1
  - Step 2: swap (2, 7) ⇒ 8 7 3 4 5 6 2 1
  - Step 3: swap (3, 6) ⇒ 8 7 6 4 5 3 2 1
  - Step 4: swap (4, 6) ⇒ 8 7 6 5 4 3 2 1
    - Stop after n/2 steps

Practice: Reverse in place



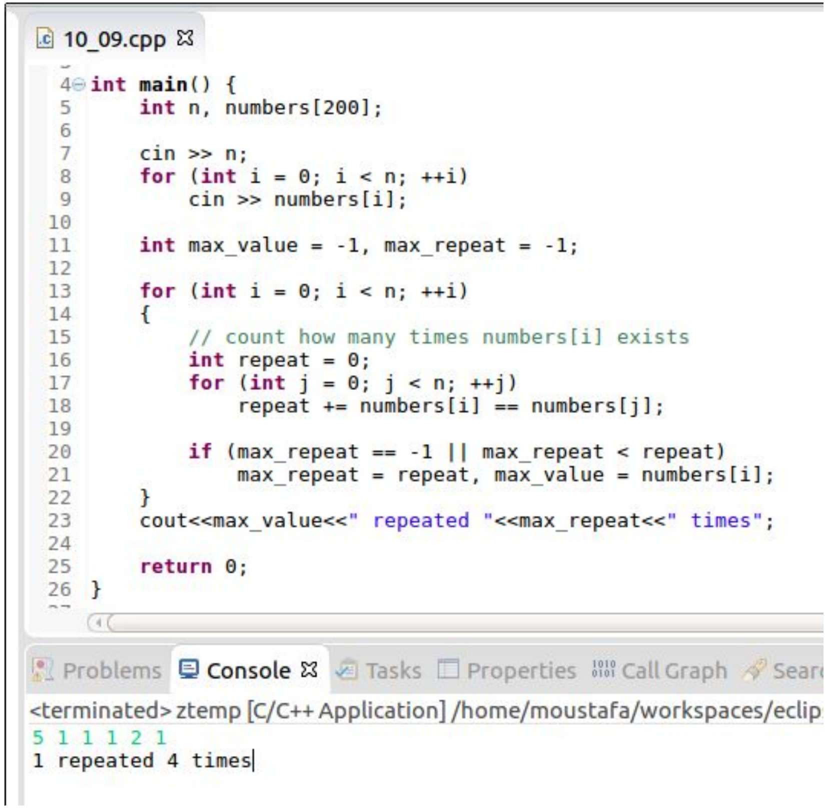
```cpp
#include<iostream>
using namespace std;

int main() {
    int n, numbers[200];

    cin >> n;
    for (int i = 0; i < n; ++i)
        cin >> numbers[i];

    for (int i = 0; i < n/2; ++i) {
        int last = n - i - 1;
        // swap positions: i and last
        int temp = numbers[i];
        numbers[i] = numbers[last];
        numbers[last] = temp;
    }

    for (int i = 0; i < n; ++i)
        cout<<numbers[i]<<" ";
    return 0;
}
```

```
<terminated> ztemp [C/C++ Application] /home/moust
6
1 2 3 4 5 6
6 5 4 3 2 1
```

Practice: Find most frequent number

- Read an Integer N, then read N <= 200 integers. Find the value that repeated the most number of times.
    - Each integer is 0 <= integer <= 150
- Example for array: 1 2 1 3 1 5 5
    - 1 repeated 3 times: the largest
    - 2 repeated 1 time
    - 5 repeated 2 times
- Stop video and think

Practice: Find most frequent number

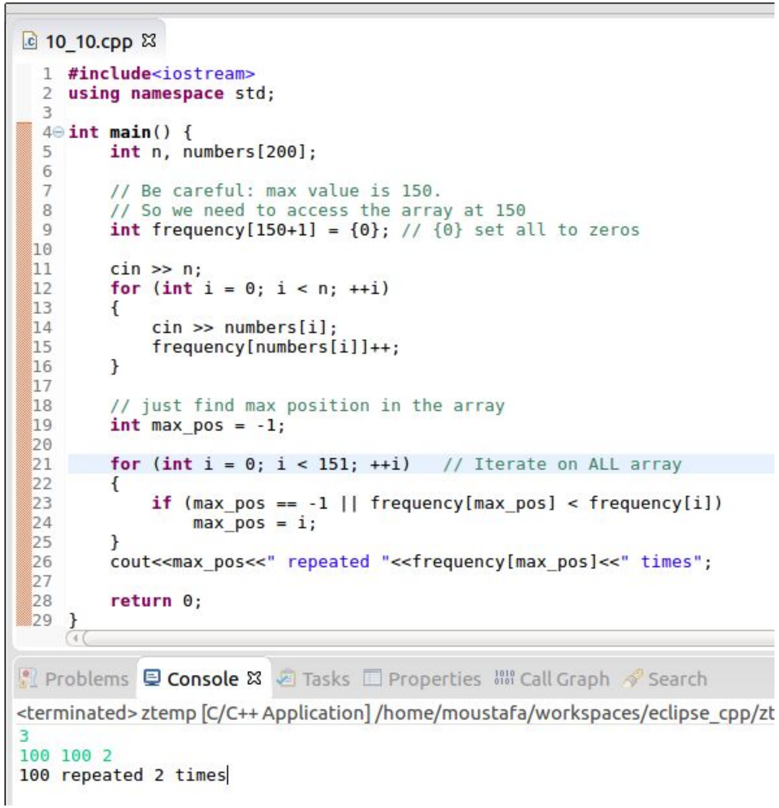

10_09.cpp

```cpp
int main() {
    int n, numbers[200];

    cin >> n;
    for (int i = 0; i < n; ++i)
        cin >> numbers[i];

    int max_value = -1, max_repeat = -1;

    for (int i = 0; i < n; ++i)
    {
        // count how many times numbers[i] exists
        int repeat = 0;
        for (int j = 0; j < n; ++j)
            repeat += numbers[i] == numbers[j];

        if (max_repeat == -1 || max_repeat < repeat)
            max_repeat = repeat, max_value = numbers[i];
    }
    cout<<max_value<<" repeated "<<max_repeat<<" times";

    return 0;
}
```

```
<terminated> ztemp [C/C++ Application] /home/moustafa/workspaces/eclip
5 1 1 1 2 1
1 repeated 4 times
```

- One easy idea
- For each number, count how many times it in the array. Find maximum of them
- Disadvantage: nested loops (much processing)
- Can you do in 1 loop only?
    - Hint: use another array

Practice: Find most frequent number - FASTER



10_10.cpp

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

int main() {
	int n, numbers[200];

	// Be careful: max value is 150.
	// So we need to access the array at 150
	int frequency[150+1] = {0}; // {0} set all to zeros

	cin >> n;
	for (int i = 0; i < n; ++i)
	{
		cin >> numbers[i];
		frequency[numbers[i]]++;
	}

	// just find max position in the array
	int max_pos = -1;

	for (int i = 0; i < 151; ++i)	// Iterate on ALL array
	{
		if (max_pos == -1 || frequency[max_pos] < frequency[i])
			max_pos = i;
	}
	cout<<max_pos<<" repeated "<<frequency[max_pos]<<" times";

	return 0;
}
```

```
<terminated> ztemp [C/C++ Application] /home/moustafa/workspaces/eclipse_cpp/zt
3
100 100 2
100 repeated 2 times
```

- Let’s use another array
- We will use a trick called frequency array
  - We think of the index as value
  - If we have M values, create array of M+1 values
- Iterate on the array and increment each time you meet a number
- Find max in the array

Run time error: Index out of boundary

- One of the most errors we do
- You access array with
  - Negative index
  - Index > its max value
- E.g. int arr[100];
- Don’t
  - arr[**100**] ⇒ Only 0 to 99
  - arr[-10]
  - The program may **crash**

Other Data types

- We focused on integer, but we can define array of other values
- **double** salary[100];
    - Array of 100 salaries
- **char** letters[300];
    - Array of 300 letters
- **string** names[200];
    - Array of 200 names

Homework 1: Search for a number

- Read an Integer N, then read N <= 200 integers [0 <= A[i] <= 500].
    - We will search in this array for numbers
- Then read integer Q (for a number of queries), then read Q integers
    - For each integer, find the **last occurance** in the array. Print its index
    - If doesn't exist, print -1
- Input 5 1 2 7 3 7 3 7 9 2
    - Means Array of 5 numbers (1 2 7 3 7) and 3 queries (7 9 2)
- Output
    - 4 [7 exists in 2 positions (2 and 4). The last is 4)
    - -1 [9 doesn't exist)
    - 1 [2 exists only in position 1]
- Easy with nested loops. Can you do with 1 loop?

Homework 2: Is increasing array?

- Read an Integer N, then read N <= 200 integers. Print YES if the array is increasing. Array is increasing if every element is >= the previous number
- Inputs
    - 4 1 2 2 5   ⇒ YES
    - 5 1 0 7 8 9 ⇒ NO   [0 is < 1, the previous number]
    - 2 -10 10 ⇒ YES

Homework 3: Replace MinMax

- Given a number N and an array A of N numbers. Assume all values [0, 2000]
- Print the array after doing the following operations:
    - Find minimum number in these numbers.
    - Find maximum number in these numbers.
    - Replace **each** minimum number with maximum number and Vise Versa.
- Input ⇒ Output
    - 7   4 1 3 10 8 10 10 ⇒ 4 10 3 1 8 1 1

Homework 4: Find the 3 minimum values

- Read integer N (>= 3), then read N integers. Find the 3 lowest numbers.
  - Don’t change the array content
  - Don’t iterate on the array more than once
- Input ⇒ Output
  - 5   4 1 3 10 8 ⇒ 1 3 4
  - 3   7 9 -2 ⇒ -2 7 9

Homework 5: Smallest pair

- Given a number N (<= 200) and an array A of N numbers.
- Print the smallest possible result of **Ai + Aj + j - i** , where 1 ≤ i < j ≤ N.
- Input ⇒ Output
  - 4   20 1 9 4   ⇒   7

Homework 6: Is Palindrome?

- Given a number N and an array A of N numbers. Determine if it's palindrome or not.
- *An array is called palindrome if it reads the same backward and forward*
    - *for example, arrays { 1 } and { 1,2,3,2,1 } are palindrome*
    - *while arrays { 1,12 } and { 4,7,5,4 } are not.*
- Inputs ⇒ Outputs
    - 5 1 3 2 3 1 ⇒ YES
    - 4 1 2 3 4 ⇒ NO

Homework 7: Find most frequent number

- Read an Integer N, then read N <= 200 integers. Find the value that repeated the most number of times.
    - Each integer is **-500** <= integer <= 270
- Example for array: 7   -1 2 -1 3 -1 5 5
    - -1 repeated 3 times: the largest
- Don’t use nested loops

Homework 8: Digits frequency

- Read an Integer N, then read N <= 200 integers. For all the digits from 0 to 9, we want to know how many times appeared
  - Input 2 78 307
  - Output:
  - 0 1
  - 1 0 [digit 1 never appeared]
  - 2 0
  - 3 1
  - 4 0
  - 5 0
  - 6 0
  - 7 2 [digit 7 appeared twice]
  - 8 1
  - 9 0

Homework 9: Recamán's sequence

- Sequence is a series of numbers. The first terms are 0, 1, 3, 6, 2, **7**, …
  - So last term **value** is 7 and its **index** is 5 (zero based)
  - The next value is either:
    - **Last value-last index-1** if 2 conditions satisfied
      - It is > 0
      - It did not appear before
      - E.g. 7 (last value) - last index (5) - 1 = 7-5-1 = 1   (> 0 but already exists
    - Or last **value+last index+1** = 7+5+1 = 13
- Read integer index ([0, 200]) and print the value of this index
  - E.g. (6 ⇒ 13), (9 ⇒ 21), (17 ⇒ 25)
- Don’t use nested loops
- The series is: 0, 1, 3, 6, 2, 7, **13**, 20, 12, **21**, 11, 22, 10, 23, 9, 24, 8, **25**, 43

Homework 10: Fixed sliding window

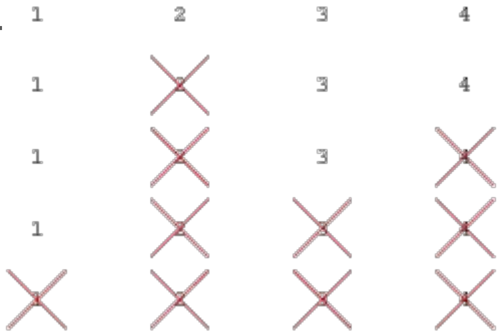
- Read Integers K and N, (where K <= N). then read N <= 200 integers. Then find a sub-array of K elements that has maximum sum.
- Input 3 7 1 0 3 -4 2 -6 9
  - Let’s list all sub-arrays of length 3
  - 1 0 3 ⇒ sum = 4
  - 0 3 -4 ⇒ sum = -1
  - 3 -4 2 ⇒ sum = 1
  - -4 2 -6 ⇒ sum = -8
  - 2 -6 9 ⇒ sum = 5
- Output: 4 6 5 (Sub-array from indices 4 to 6 has maximum sum of 5)
- Hard: Can you do without nested loops? There are 2 ways.

Homework 11: Count increasing subarrays

- Read an Integer N, then read N <= 200 integers. Count how many sub-arrays are increasing in the array. A sub-array is set of consecutive numbers in array
- E.g. If array is 1 2 3 4
  - We can find all sub-arrays of length 1 ⇒ 1 / 2 / 3 / 4
  - All sub-arrays of length 2 ⇒ 1, 2 / 2, 3 / 3, 4
  - All sub-arrays of length 3 ⇒ 1, 2, 3 / 2, 3, 4
  - All sub-arrays of length 4 ⇒ 1, 2, 3, 4
- Inputs ⇒ Outputs
  - 4 1 2 3 4 ⇒ 10 [10 sub-arrays from previous example, all are increasing]
  - 4 4 3 2 1 ⇒ 4 [only sub-arrays of length 1 can be considered]
  - 4 10 20 1 5 ⇒ 6
- Easy using 3 nested loops. Medium using 2 loops. Can you do it with 1 loop?

Homework 12: Josephus problem

- Read integers N (<= 200) and K (<= 1000000). Find the game winner for following game:
- We have a group of N people in Circle. They are numbered 1, 2, …. N
  - Someone is the master of the game.
  - He starts from Person #1. Count K. Then remove this person from the circle.
  - He keeps doing so till only 1 person remains. This is the winner.
- Input 4 2
  - Means we have people: 1, 2, 3, 4. Master starts at 1
  - Count 2 persons (2 removed), start from 3
  - Count 2 persons (4 removed), start from 1
  - Count 2 persons (3 removed), 1 is winner
- Output
  - People removed in order: 2 4 3 1 [same answer for 10 2 why?]



Homework 13: longest subarray

- Read integers N (<= 1000) then N numbers each is either 0 or 1. Find longest **subarray** with number of zeros = numbers of ones
  - You can easily implement it using 3 loops
  - Or with little thinking using 2 loops (even with no extra arrays)
  - Hard: You can implement it without any nested loops
- Inputs ⇒ outputs
  - 7 1 0 0 0 1 1 1 ⇒ 6 (e.g. 100011 or 000111)
  - 19 1 0 0 0 0 0 1 0 1 1 0 1 0 0 0 0 0 0 1 ⇒ 8 (e.g. 00101101)
- Reduction
  - How may this problem be reduced to another problem: longest subarray of zero sum?

تم بحمد الله

علمكم الله ما ينفعكم

ونفعكم بما تعلمتم

وزادكم علماً