Міністерство освіти і науки, молоді та спорту України

Національний технічний університет України

«Київський політехнічний інститут»

[Алгоритми та структури даних](http://wiki.kpi.ua/index.php/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC%D0%B8%20%D1%82%D0%B0%20%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B8%20%D0%B4%D0%B0%D0%BD%D0%B8%D1%85_(15501550))

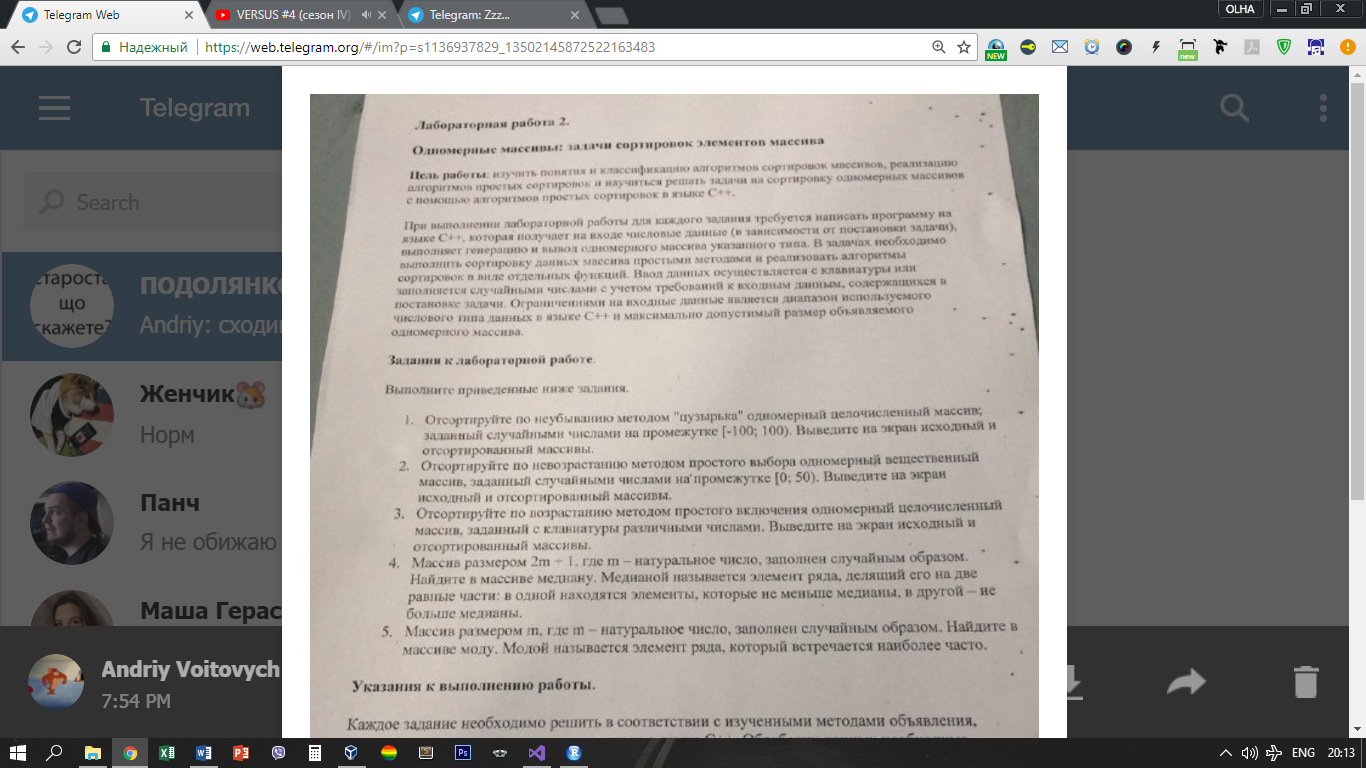
**ЗВІТ ДО**

**ЛАБОРАТОРНОЇ РОБОТИ № 2**

Варіант № 1

|  |  |
| --- | --- |
| Оцінка «\_\_\_\_\_\_\_\_\_» | Виконав: студент 2 курсу  гр.ТВ-61  Артамонов Олексій Юрійович |

Київ – 2017



#include "stdafx.h"

#include "stdafx.h"

#include <stdio.h>

#include <math.h>

#include <conio.h>

#include <stdlib.h>

#include <windows.h>

#include <time.h>

#include <iostream>

using namespace std;

void Arr\_Zap(int arr[10]){

cout << "Original array: ";

for(int i = 0; i < 10; i++){

arr[i] = rand() % (100 + 100 + 1) - 100;

cout << arr[i] << " ";

}

cout << endl;

}

void Bubble\_Sort(int arr[10]){

for(int i = 0; i < 10; i++){

for(int j = 9; j > 0; j--){

if(arr[j-1] >= arr[j]){

int tmp = arr[j-1];

arr[j-1] = arr[j];

arr[j] = tmp;

}

}

}

}

void Vivod(int arr[10]){

cout << "Sorted array: ";

for(int i = 0; i < 10; i++){

cout << arr[i] << " ";

}

}

void Arr1\_Zap(double arr1[10]){

cout << "Original array: ";

for(int i = 0; i < 10; i++){

arr1[i] = rand() % (50 + 1);

cout << arr1[i] << " ";

}

cout << endl;

}

void Selection\_Sort(double arr1[10]){

for(int i = 0; i < 9; i++){

int min = i;

for(int j = i + 1; j < 10; j++){

if(arr1[j] >= arr1[min]){

min = j;

}

}

int temp = arr1[i];

arr1[i] = arr1[min];

arr1[min] = temp;

}

}

void Vivod1(double arr1[10]){

cout << "Sorted array: ";

for(int i = 0; i < 10; i++){

cout << arr1[i] << " ";

}

}

void Arr2\_Zap(int arr2[10]){

cout << "Enter original array: \n";

for(int i = 0; i < 10; i++){

cin >> arr2[i];

}

cout << endl;

}

void Insert\_Sort(int arr2[10]){

for (int i = 0; i < 10; i++) {

int tmp = arr2[i]; //поиск места элемента

int j = i;

while((j > 0) && (arr2[j - 1] > tmp)){

arr2[j] = arr2[j - 1];

j--;

}arr2[j] = tmp;

}

}

void Vivod2(int arr2[10]){

cout << "Sorted array: ";

for(int i = 0; i < 10; i++){

cout << arr2[i] << " ";

}

}

void Arr3\_Zap(int m, int arr3[]){

for(int i = 0; i < 2 \* m + 1; i++){

arr3[i] = rand() % 10;

}

cout << endl;

}

void Median(int m, int arr3[]){

for(int i = 0; i < 2\*m+1; i++){

int min = i;

for(int j = i + 1; j < 2\*m+1; j++){

if(arr3[j] > arr3[min]){

min = j;

}

}

int temp = arr3[i];

arr3[i] = arr3[min];

arr3[min] = temp;

}

cout << endl << endl << "\nArray: ";

for(int i = 0; i < 2 \* m + 1; i++){

cout << arr3[i] << " ";

}

cout << "\nMedian:" << arr3[m] << endl;

}

void Arr4\_Zap(int m, int arr4[]){

cout << endl << endl << "Array: ";

for(int i = 0; i < m; i++){

arr4[i] = rand() % 10;

cout << arr4[i] << " ";

}

}

void Mod(int m, int arr4[]){

int ch = 0,

ch\_m = 0,

mod = 0,

max\_mod = 0,

flag = 0;

for (int i = 0; i < m; i++){

ch = 0;

for(int j = 0; j < m; j++){

if(arr4[i] == arr4[j]){

ch++;

}

}

if(ch > max\_mod){

flag = 1;

max\_mod = ch;

ch\_m = arr4[i];

}

else if(ch == max\_mod && arr4[i] != ch\_m){

flag = 0;

}

mod = ch\_m;

}

cout << "\nModa: " << mod << "\n\n";

}

int main()

{

srand(time(NULL));

const int m = 12;

cout << "Bubble Sort:" << endl;

int arr[10];// rand[-100; 100), pyzurek

Arr\_Zap(arr);

Bubble\_Sort(arr);

Vivod(arr);

cout << "\n\nSelection Sort:" << endl;

double arr1[10];//rand[0; 50), prostoy vibor

Arr1\_Zap(arr1);

Selection\_Sort(arr1);

Vivod1(arr1);

cout << "\n\nInsert Sort:" << endl;

int arr2[10];//klava, prosyoe vklyuchenie

Arr2\_Zap(arr2);

Insert\_Sort(arr2);

Vivod2(arr2);

int arr3[2 \* m + 1];//rand, <=|>=

Arr3\_Zap(m, arr3);

Median(m, arr3);

int arr4[m];//rand, nayti samiy chastiy el

Arr4\_Zap(m, arr3);

Mod(m, arr3);

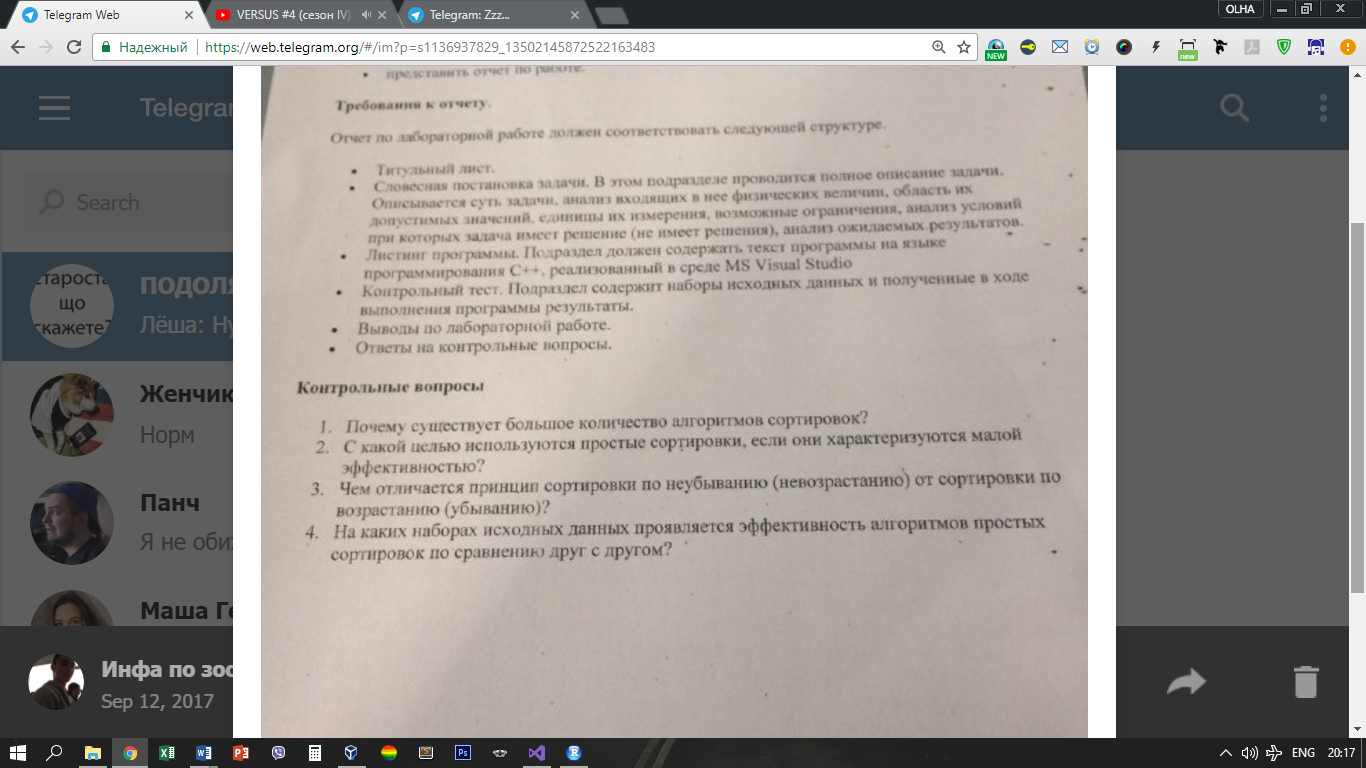
\_getch();

return 0;

}

Выводы:

В данной лабораторной работе были реализованы различные методы элементарных сортировок и показано как использовать их в прикладных задачах по поиску моды и медианы выборки.



1.Потому что каждый алгоритм был создан для своих специфических задач. Одни алгоритмы хороши для больших объемов, другие - для почти отсортированных данных, третьи, когда есть много памяти или когда памяти слишком мало.

2. Простые методы удобны для объяснения принципов сортировок, т.к. имеют простые и короткие алгоритмы.

3. Если все элементы массива различны, то надо говорить о убывающем (или возрастающем) порядке. Иначе речь идет о неубывающем (или невозрастающем) порядке.

4. Для небольших массивов простые методы более эффективны, для больших объемов данных- более эффективны усложненные методы.

