**МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ ФЕДЕРАЦИИ**  
Федеральное государственное бюджетное образовательное учреждение высшего образования  
**«РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА (МИИТ)»  
 РУТ (МИИТ)**  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
Кафедра: «Вычислительные системы и сети»  
  
  
  
  
  
  
  
  
  
  
**О Т Ч Ё Т  
П О Л А Б О Р А Т О Р Н О Й Р А Б О Т Е № 1**  
по дисциплине «Структуры и алгоритмы обработки данных»  
на тему: «Алгоритмы сортировок и поиска данных»  
  
  
  
  
  
  
  
  
  
 Выполнил: студент группы УИБ-311   
 Владыка П.А.

Москва 2018

**Задание**

Первая лабораторная работа посвящена методам сортировки данных в списке. Цель работы: изучение основных методов сортировки и поиск методом половинного деления.

Разработать и отладить программу, реализующую:

1. Ввод исходного списка (размерность массива задаётся пользователем).
2. Сортировку списка методом «пузырька», ввести переменную-флаг, подчитать количество перестановок.
3. Сортировку списка с помощью метода «экстремумов». Подсчитать количество сравнений.
4. Сортировку массива «методом простой вставки», применяя поиск методом половинного деления.
5. Меню для выбора варианта сортировки.

**Таблица имён**

|  |  |
| --- | --- |
| **Исходные данные** | |
| numbers | Исходный генерируемый массив чисел |
| **Промежуточные данные** | |
| size | Размер массива, вводимый с клавиатуры |
| numbers2 | Промежуточная копия массива для сброса сортировки |
| log | Логическая переменная, используемая в функции |
| d, save1, save2 | «ячейка» обмена |
| i, j, k, y | счётчики |

**Блок-схема**

**Главная программа Конструктор класса** INCREASE



**Исходный код программы**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<iostream>

#include<stdlib.h>

#include<stdio.h>

#include<string.h>

#include<time.h>

#include<cstdlib>

#include<conio.h>

#include<windows.h>

#include <iomanip>

#define FOUND(i) makeResult(i, 1)

#define NOTFOUND(i) makeResult(i, 0)

using namespace std;

class INCREASE{

public:

INCREASE() {

cout << "Введите размер массива ";

cin >> size;

numbers = new int[size];

numbers2 = new int[size];

for (int i = 0; i < size; i++)

numbers[i] = rand() % 200 + 1;

memcpy(numbers2, numbers, 4 \* size);//Копирование сгенерированного массива чисел (сброс сортировки) }

~INCREASE() {delete[] numbers, numbers2;}

enum ConsoleColor {

Black = 0,

Yellow = 14,

White = 15,

Red = 4,

Blue = 1};

int\* numbers, \*numbers2; /\*Массив для сортировки\*/

int size;

void menu(){

char c; /\*номер опции\*/

SetColor(White, Blue);

system("cls");

cout << "{{>================================================<}}\n"

<< "|| Сортировка по возрастанию ||\n"

<< "{{>================================================<}}\n"

<< "|| 1. Сортировать методом \"Пузырька\" ||\n"

<< "|| 2. Сортировать методом \"Экстремумов\" ||\n"

<< "|| 3. Сортировать методом простой вставки ||\n"

<< "|| при помощи метода половинного деления ||\n"

<< "|| 4. Просмотреть массив ||\n"

<< "|| 5. Сбросить сортировку ||\n"

<< "|| 6. Очистить массив ||\n"

<< "|| 7. Выход ||\n"

<< "{{>================================================<}}\n"

<< "\nВыбранная опция: ";

(cin >> c).get();

switch (c) {

case 49: {

Bubble();

break;}

case 50: {

Extremum();

break;}

case 51: {

Insert();

break;}

case 52: {

Check();

break;}

case 53: {

Reset();

break;}

case 54: {

STZ();

break;}

case 55: {

break;}

default: menu();}}

private:

void Bubble() {

system("cls");

bool log = false;

int d;

for (;;) {

for (int i = 0; i < size - 1; i++) {

if (numbers2[i] > numbers2[i + 1]) {

d = numbers2[i + 1];

numbers2[i + 1] = numbers2[i];

numbers2[i] = d;

log = true;}}

if (log != true) {

break;}

else {

log = false;}}

cout << "ОТСОРТИРОВАНО!" << endl;

\_getch();

menu();}

void Extremum() {

system("cls");

int d = 0;

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

d = numbers2[j];

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

if (d < numbers2[i]) {

d = numbers2[i];

numbers2[i] = numbers2[j];

numbers2[j] = d;}}}

cout << "ОТСОРТИРОВАНО!" << endl;

\_getch();

menu();}

struct Result {

size\_t pos;

int isFound;};

struct Result makeResult(size\_t pos, int isFound) {

struct Result r;

r.pos = pos;

r.isFound = isFound;

return r;}

struct Result binarySearch(size\_t s, int a[], int isk) {

size\_t first = 0;

size\_t lastp = s;

if (s == 0) {return NOTFOUND(0);}

else if (a[0] > isk) {return NOTFOUND(0);}

else if (a[s - 1] < isk) {return NOTFOUND(s);}

while (first < lastp) {

size\_t mid = first + (lastp - first) / 2;

Out(mid, s, Yellow);

if (isk <= a[mid])

lastp = mid;

else first = mid + 1;}

if (a[lastp] == isk) {

return FOUND(lastp);}

else {return NOTFOUND(lastp);}}

void Out(int position, int max, enum ConsoleColor col) {

system("cls");

cout << "Ваш массив:" << endl;

for (int y = 0; y < max; y++) {

if (y == position) {

SetColor(col, Black);

if (y % 20 == 0)

cout << endl;

cout << setw(4) << numbers2[y] << " ";

SetColor(White, Black);}

else {

if (y % 20 == 0)

cout << endl;

cout << setw(4) << numbers2[y] << " ";}}

Sleep(100);}

void Insert() {

SetColor(White, Black);

system("cls");

int Save1, Save2, k = 0, c = 0;

numbers2[0] = numbers[0]; //Вставка первого элемента в массив

struct Result r;

for (int i = 1; i < size; i++) {

r = binarySearch(i, numbers2, numbers[i]);

Save1 = numbers2[r.pos]; //Сохранение элемента

Out(r.pos, i, Red);

numbers2[r.pos] = numbers[i];

for (k = r.pos; k < size - 1; k++) {

Save2 = numbers2[k + 1];

numbers2[k + 1] = Save1;

Save1 = Save2;}

Out(r.pos, i + 1, Red);}

cout << "ОТСОРТИРОВАНО!" << endl;

\_getch();

menu();}

void Check() {

system("cls");

cout << "Ваш массив:" << endl;

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

if (i % 20 == 0)

cout << endl;

cout << setw(4) << numbers2[i] << " ";}

\_getch();

menu();}

void SetColor(ConsoleColor text, ConsoleColor background) {

HANDLE hStdOut = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hStdOut, (WORD)((background << 4) | text));}

void STZ() {

system("cls");

memset(numbers2, 0, 4 \* size);

cout << "МАССИВ ОЧИЩЕН!" << endl;

\_getch();

menu();}

void Reset() {

system("cls");

memcpy(numbers2, numbers, 4 \* size);//Копирование сгенерированного массива чисел (сброс сортировки)

cout << "СОРТИРОВКА СБРОШЕНА!" << endl;

\_getch();

menu();}};

int main(){

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

srand(time(0));

INCREASE \*inc = new INCREASE();

cout << "Сгенерированный массив:" << endl;

for (int i = 0; i < inc->size; i++) {

if (i % 20 == 0)

cout << endl;

cout << setw(4) << inc->numbers[i] << " ";}

\_getch();

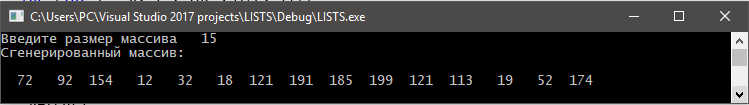
inc->menu();

delete inc;

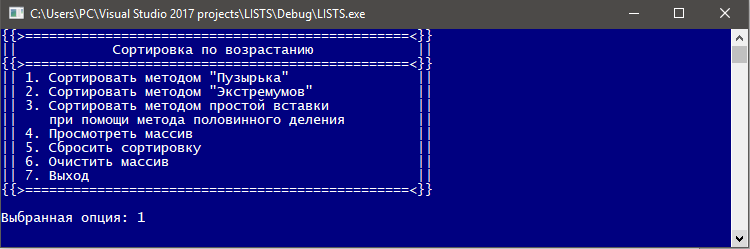
return 0;}

**Отладка**

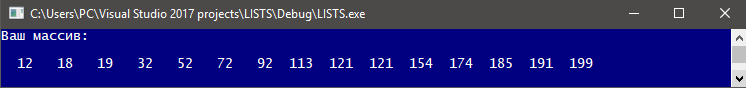
Ввод размера массива чисел и генерация массива этого размера:



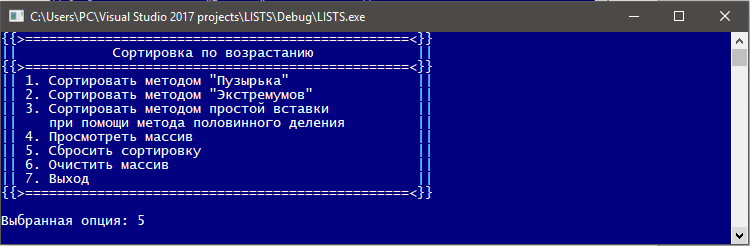
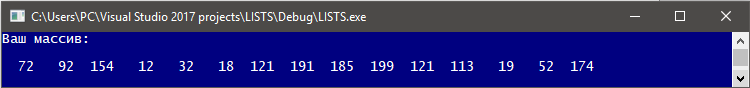
Сортировка методом «Пузырька»:



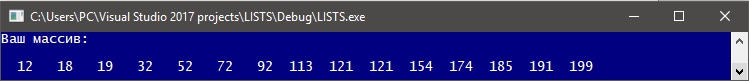
Просмотр отсортированного массива:



Сброс сортировки:

Сортировка методом «Экстремумов»:



Сортировка методом простой вставки (требует удаления массива, элементы берутся из копии исходного массива):

