Министерство образования и науки РФ

СевГУ

Кафедра ИС

Отчёт

по лабораторной работе №6

«Анализ алгоритмов сортировки»

Выполнил:

ст.гр. ИТ-12д

Воронин И.Ю.

Проверил:

Заикина Е.Н.

Севастополь

2015

**1. ЦЕЛЬ РАБОТЫ**

Научиться оценивать сложность и количество операций для алгоритмов сортировки.

**2. ВАРИАНТЫ ЗАДАНИЙ**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| В | **Простейший алгоритм 1** | **Простейший алгоритм 2** | **Быстрый алгоритм 3** | **Структура и критерий**  **сортировки** |
| 4 | Сортировка посредством выбора | Модифицированный  метод «пузырька» | Быстрая сортировка | Структура Студент, содержит ФИО, курс, факультет, сортировка по убыванию курса |

**3.ТЕКСТ ПРОГРАММЫ**

#include <conio.h>

#include <Windows.h>

#include <iostream>

using namespace std;

//основная структура данных

struct student{

char FIO[10], fackult[10];

unsigned short kurs;

};

extern void quickSort(student \*a, int n);

extern void bubbleSort(student \*a, int n);

extern void selectSort(student \*a, int n);

int main(){

int N;

unsigned long long t1,t2,t3;

setlocale(LC\_ALL,"Russian");

cout<<"Введите чило элементов массива: ", cin>>N;

student a[N]; //одномерный массив

//БЫСТРАЯ СОРТИРОВКА

cout<<endl<<endl<<endl<<"Быстрая сортировка: "<<endl;

for(int i=0;i<N;i++) a[i].kurs=rand()%5+1;

t1=GetTickCount();

cout<<endl<<"До: "<<endl;

for(int i=0;i<N;i++) cout<<a[i].kurs<<' ';

quickSort(a,N);

cout<<endl<<endl<<"После: "<<endl;

for(int i=0;i<N;i++) cout<<a[i].kurs<<' ';

cout<<endl<<endl<<"Потрачено времени: "<<(GetTickCount()-t1)<<" мc."<<endl;

//СОРТИРОВКА ПУЗЫРЬКОМ

cout<<endl<<endl<<endl<<"Сортировка пузырьком : "<<endl;

for(int i=0;i<N;i++) a[i].kurs=rand()%5+1;

t2=GetTickCount();

cout<<endl<<"До: "<<endl;

for(int i=0;i<N;i++) cout<<a[i].kurs<<' ';

bubbleSort(a,N);

cout<<endl<<endl<<"После: "<<endl;

for(int i=0;i<N;i++) cout<<a[i].kurs<<' ';

cout<<endl<<endl<<"Потрачено времени: "<<(GetTickCount()-t2)<<" мc."<<endl;

//СОРТИРОВКА ВЫБОРОМ

cout<<endl<<endl<<endl<<"Сортировка выбором: "<<endl;

for(int i=0;i<N;i++) a[i].kurs=rand()%5+1;

t3=GetTickCount();

cout<<endl<<"До: "<<endl;

for(int i=0;i<N;i++) cout<<a[i].kurs<<' ';

selectSort(a,N);

cout<<endl<<endl<<"После: "<<endl;

for(int i=0;i<N;i++) cout<<a[i].kurs<<' ';

cout<<endl<<endl;

cout<<"Потрачено времени: "<<(GetTickCount()-t3)<<" мc."<<endl;

return 0;

}

void quickSort(student \*a, int n) {

int i = 0, j = n; // поставить указатели на исходные места

int temp, p;

p = a[n/2].kurs; // центральный элемент

// процедура разделения

do {

while (a[i].kurs < p)

i++;

while (a[j].kurs > p)

j--;

if (i <= j) {

swap(a[i], a[j]);

i++;

j--;

}

}

while(i <= j);

// рекурсивные вызовы, если есть, что сортировать

if(j > 0)

quickSort(a, j);

if(n > i)

quickSort(a+i, n-i);

}

void bubbleSort(student \*a, int n) {

int i, j;

for(i = n; i > 1; i--)

for(j = 1; j < i; j++) // выталкивается самый тяжелый

if(a[j].kurs > a[j+1].kurs) // если он больше своего соседа справа

swap(a[j], a[j+1]);

}

void selectSort(student \*a, int n) {

int i, j, imax;

for(i = n; i > 1; i--) { // заполнение i-й ячейки

imax = 1;

for(j = 1; j <= i; j++) // поиск максимального элемента

if(a[j].kurs > a[imax].kurs)

imax = j;

swap(a[i], a[imax]);//вставка максимального элемента на свое место

}

}

**4.ОЦЕНКА АЛГОРИТМОВ**

O(n \log n)Оценка алгоритма быстрой сортировки представлена в виде Худшее время равно O(*n*2), а лучшее O(*n* log *n*). Общая сложность алгоритма определяется лишь количеством разделений, то есть глубиной рекурсии. Глубина рекурсии, в свою очередь, зависит от сочетания входных данных и способа определения опорного элемента.

Оценка алгоритма сортировки пузырьком представлена в виде O(*n*2). Худшее время равно O(*n*2), а лучшее O(*n*). Проходы по массиву повторяются  N-1  раз или до тех пор, пока на очередном проходе не окажется, что обмены больше не нужны, что означает — массив отсортирован.

Оценка алгоритма сортировки прямого выбора представлена в виде O(*n*2) и в среднем, и в лучшем, и в худшем случаях. [Затраты памяти](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%81%D0%BB%D0%BE%D0%B6%D0%BD%D0%BE%D1%81%D1%82%D1%8C) О(*n*) всего, O(1) дополнительно.

**5.АНАЛИЗ РАБОТЫ ПРОГРАММЫ**

Запуск программы для 100 элементов массива выглядит следующим образом:

* сортировка пузырьком (рис.5.1)

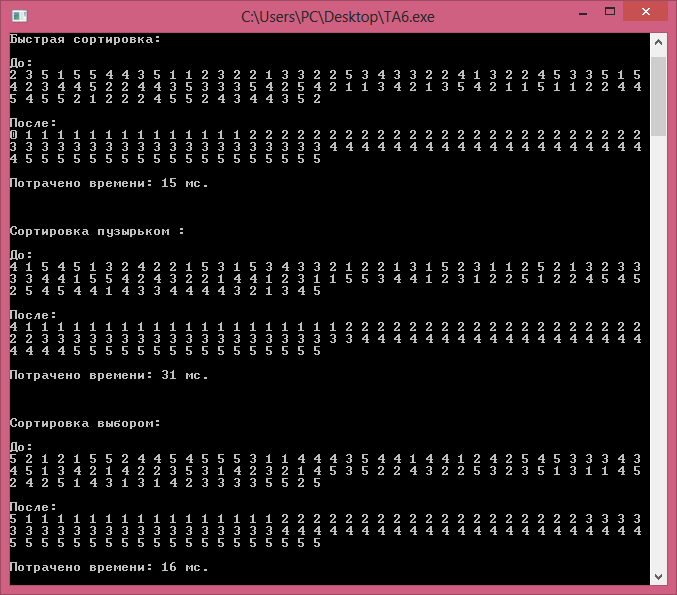


Рисунок 5.1-Результат работы сортировки пузырьком.

* сортировка выбором (рис.5.2)

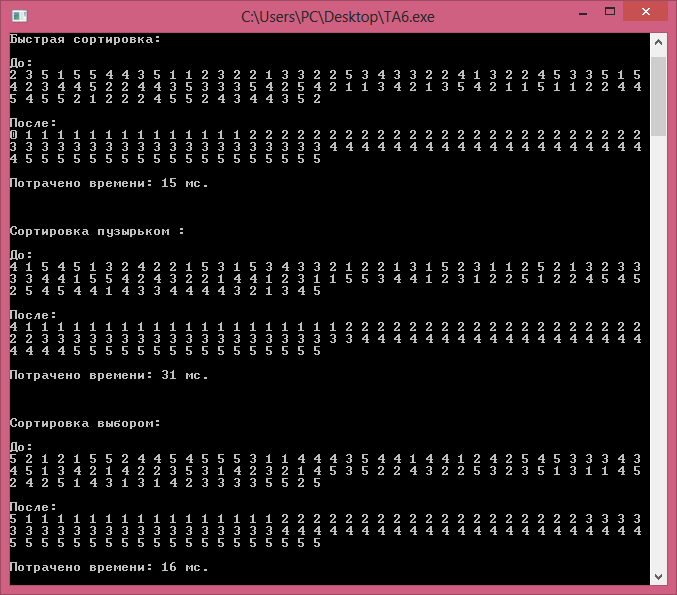


Рисунок 5.2-Результат работы сортировки выбором.

* быстрая сортировка (рис.5.3)

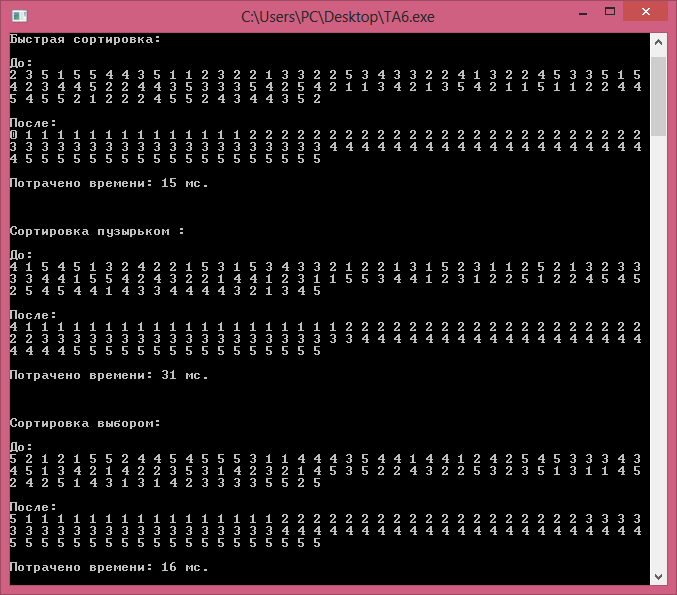


Рисунок 5.3-Результат работы быстрой сортировки.

ВЫВОДЫ

В данной работе были проанализированы особенности алгоритмов сортировки одномерных массивов. Самым не рациональным оказался алгоритм сортировки пузырьком, так как он затрачивает большое число времени и производит большое число итераций. Однако на его основе был придуман алгоритм быстрой сортировки, который быстрее двух друг отсортировал массив в данном тесте.

Однако в зависимости от исходных данных результаты могут варьироваться от худшего случая к лучшему.