Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное   
учреждение высшего образования

Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского

Институт информационных технологий, математики и механики

**Отчет по лабораторной работе**

**«Моделирование работы сортировщика файлов ОС»**

**Выполнил**:

студент группы 381803-1

Сермяжко Е. А.

**Проверил**:

доцент кафедры МОСТ, к.т.н.,

Сысоев А.В.

Нижний Новгород

2018

**Содержание**

[Введение 3](#_Toc530777841)

[1. Постановка задачи 4](#_Toc530777842)

[2. Руководство пользователя 5](#_Toc530777843)

[3.1. Описание структуры программы 7](#_Toc530777844)

[3.2. Описание алгоритмов 7](#_Toc530777845)

[4. Результаты экспериментов 11](#_Toc530777846)

[Заключение 12](#_Toc530777847)

[Литература 13](#_Toc530777848)

[Приложение 14](#_Toc530777849)

# Введение

Практически в каждом программном проекте возникает необходимость обработки большого числа единообразно организованных данных. Одним из важнейших способов обработки массивов является сортировка. В наше время для решения сложных задач в программировании необходимо работать с большими объёмами информации, куда выгоднее сначала упорядочивать используемый массив данных в нужном порядке, что значительно снизит затраты во времени при поиске какого-либо элемента из этого массива.

# Постановка задачи

Разработать прототип файлового менеджера с функцией показа файлов в заданном каталоге, упорядоченных по возрастанию/убыванию размера.

Входные данные:

* Путь до директории, в которой необходимо отсортировать содержимое.
* Метод сортировки.

Выходные данные:

* Отсортированный список имен файлов с указанием размера.
* Время сортировки.

Программа должна предоставлять пользователю возможность сменить метод сортировки и повторно формировать выходные данные.

Программа должна реализовывать диалог с пользователем посредством интерфейса, который включает:

* возможность ввода пути до заданного каталога;
* возможность выбора метода сортировки;
* возможность просмотра отсортированного списка файлов с указанием размера.

# Руководство пользователя

В начале программы пользователь должен ввести путь до директории (см. пример на Рис. 1.)



Рис. 1. Начало программы

При неправильном вводе данных программа оповещает пользователя об ошибке и предлагает ввести путь заново.



Рис. 2. Сообщение об ошибке

После правильного ввода пути программа выводит названия, размер и количество файлов, которые находятся в заданной директории. Далее пользователю предоставляется выбор сортировки.

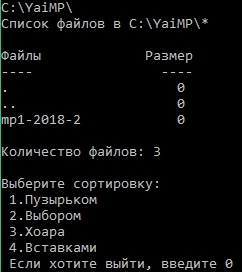


Рис. 3. Выбор сортировки

Если введена цифра, не соответствующая инструкции, программа сообщает об ошибке и предлагает ввести номер сортировки заново.

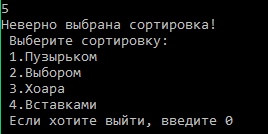


Рис. 4. Ошибка ввода сортировки

После верного ввода программа сортирует файлы и выводит их на экран в упорядоченном по возрастанию виде с указанием времени. После предлагается вновь выбрать вид сортировки.

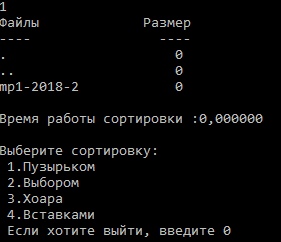


Рис. 5. Выполнение сортировки

Программа закончит свою работу только после нажатия 0.

1. **Руководство программиста**

## Описание структуры программы

Программа написана на языке программирования СИ. Состоит из одного файла Source.cpp.

В программе реализованы следующие функции:

**1. Функция "Bub"**

Сортировка пузырьком.

Принимает на вход массив типа unsigned long (a), два массива типа char (b) и (direct), размер массива тип long (number). Результат: упорядоченный массив.

**2. Функция "Cho"**

Сортировка выбором.

Принимает на вход массив типа unsigned long (a), два массива типа char (b) и (direct), размер массива тип long (number). Результат: упорядоченный массив.

**3. Функция "Sort"**

Сортировка Хоара.

Принимает на вход массив типа unsigned long (a), два массива типа char (b) и (direct), размер массива тип long (number). Результат: упорядоченный массив.

**4. Функция "Ins"**

Сортировка вставками.

Принимает на вход массив типа unsigned long (a), два массива типа char (b) и (direct), размер массива тип long (number). Результат: упорядоченный массив.

## Описание алгоритмов

Сортировки, используемые в программе:

**1. Пузырьком:**

Идея метода: шаг сортировки состоит в проходе снизу вверх по массиву. По пути просматриваются пары соседних элементов. Если элементы некоторой пары находятся в неправильном порядке, то меняем их местами. После нулевого прохода по массиву "вверху" оказывается самый "легкий" элемент - отсюда аналогия с пузырьком. Следующий проход делается до второго сверху элемента, таким образом второй по величине элемент поднимается на правильную позицию...

Делаем проходы по все уменьшающейся нижней части массива до тех пор, пока в ней не останется только один элемент. На этом сортировка заканчивается, так как последовательность упорядочена по возрастанию.

*Псевдокод:*

Функция Пузырек (символьный a (массив), символьный b (массив), символьный direct(массив), целый number)   
  
{   
целый i, j, puz;   
символьный buffer(массив);   
  
цикл для i от 0 до number-1 , i = i+1  
 цикл для j от number-1 до i+1 , j = j - 1  
 если (a[ j ] < a[ j-1 ]), то   
 {   
 puz = a[ j ];   
 a[ j ] = a[ j-1 ];   
 a[ j-1 ] = puz;   
  
 скопировать из b[j] в buffer   
 скопировать из b[j-1] в b[i]  
 скопировать из buffer в b[j-1]  
 }   
}

**2. Сортировка выбором:**

Идея метода состоит в том, чтобы создавать отсортированную последовательность путем присоединения к ней одного элемента за другим в правильном порядке. Ищем минимальный (максимальный) элемент в массиве, а после меняем его с первым, далее производим ту же операцию с неотсортированным массивом, т.е. следующий минимальный (максимальный) элемент меняем со 2 и т.д. Алгоритм продолжается, пока не останется один элемент массива.

*Псевдокод:*

Функция Выбором (симв a (массив), симв b (массив), симв direct(массив), цел number)   
  
{   
цел i, j, puz, vb;   
симв buffer(массив);   
цикл для i от 0 до number-1, i = i + 1  
{   
puz = i;   
vb = a[ i ];   
  
цикл для j от i+1 до number-1   
 если (a[ j ] > vb), то   
 {   
 vb = a[ j ];   
 puz = j;   
 }   
a[ puz ] = a [ i ];   
a[ i ] = vb;   
  
скопировать из b[puz] в buffer   
скопировать из b[i] в b[puz]  
скопировать из buffer в b[i]  
}   
}

**3. Сортировка Хоара:**

Метод основан на подходе "разделяй-и-властвуй". Общая схема такова:

1. Из массива выбирается некоторый опорный элемент a[i]

2. Запускается процедура разделения массива, которая перемещает все ключи, меньшие, либо равные a[i], влево от него, а все ключи, большие, либо равные a[i] - вправо

3. Теперь массив состоит из двух подмножеств, причем левое меньше, либо равно правого

4. Для обоих подмассивов: если в подмассиве более двух элементов, рекурсивно запускаем для него ту же процедуру

В конце получится полностью отсортированная последовательность.

*Псевдокод:*

Функция Хоара (симв a (массив), симв b (массив), симв direct(массив), цел number)   
  
{   
цел i = 0, j = number - 1;   
цел temp, mid;   
симв buffer  
mid = a[ number/2 ];   
  
цикл до   
{   
 цикл пока (a[ i ] < mid)   
 i = i+1;   
 цикл пока (a[ j ] > mid)   
 j = j-1;   
 если (i <= j), то   
 {   
 temp= a[ i ];   
 a[ i ] = a[ j ];   
 a[ j ] = temp;   
  
 скопировать из b[i] в buffer   
 скопировать из b[j] в b[i]   
 скопировать из buffer в b[j]  
 i = i+1;   
 j = j-1;   
 }   
}   
пока (i <= j)   
  
если (j > 0), то   
 Хоара(a, b, direct, j+1);   
  
если (number > i), то   
 Хоара(a + i, b + i, direct, number-i);   
}

**4. Сортировка вставками:**

Поиск подходящего места для элемента входной последовательности осуществляется путем последовательных сравнений с элементом, стоящим перед ним.

В зависимости от результата сравнения элемент либо остается на текущем месте(вставка завершена), либо они меняются местами и процесс повторяется.

Таким образом, в процессе вставки мы "просеиваем" элемент x к началу массива, останавливаясь в случае, когда найден элемент, меньший x или достигнуто начало последовательности.

*Псевдокод:*

Функция Вставками (симв a (массив), симв b (массив), симв direct(массив), цел number)   
  
{   
цел x;  
цел i, j;   
симв buffer(массив);   
  
цикл для i от 0 до number-1, i = i+1  
{   
 x = a[ i ];   
 j = i-1;   
 цикл пока (j >= 0) и (c < a[ j ]) , j= j - 1  
 {   
 a[ j+1 ] = a[ j ];   
  
 скопировать из b[j+1] в buffer   
 скопировать из b[j] в b[j+1]   
 скопировать из buffer в b[j]   
 }   
a[ j+1 ] = x;   
}   
}

# Результаты экспериментов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Кол-во элементов в директории | Сортировка пузырьком  сек. | Сортировка выбором  сек. | Сортировка Хоара  сек. | Сортировка вставками  сек. |
| 3143 | 0,397 | 0,017 | 0,002 | 0,686 |
| 0,621 | 0,03 | 0,002 | 0,658 |
| 0,695 | 0,026 | 0,001 | 0,639 |
| 0,655 | 0,028 | 0,005 | 0,652 |
| 0,652 | 0,019 | 0,002 | 0,64 |
| 4641 | 0,899 | 0,043 | 0,02 | 1,558 |
| 1,499 | 0,039 | 0,02 | 1,491 |
| 1,481 | 0,04 | 0,037 | 1,435 |
| 1,471 | 0,036 | 0,02 | 1,468 |
| 1,48 | 0,039 | 0,03 | 1,479 |

# Заключение

Анализируя полученные результаты, мы приходим к выводу, что сортировка "Хоара" является самой быстрой из рассматриваемых сортировок. Самыми медленными сортировками являются "пузырек" и "вставками". Сортировка "выбором" быстрее, чем "пузырек" и "вставки", но уступает сортировке "Хоара".  
  
Итог по времени:  
1. Хоара  
2. Выбором  
3. Вставками  
4. Пузырек

# Литература

1. Кнут Д. Э. Искусство программирования. Том 3. Сортировка и поиск = The Art of Computer Programming. Volume 3. Sorting and Searching / под ред. В. Т. Тертышного (гл. 5) и И. В. Красикова (гл. 6). – 2-е изд. – Москва: Вильямс, 2007. – Т. 3. – 832 с.
2. Сайт Algolist. Сортировка выбором – <http://algolist.manual.ru/sort/select_sort.php>.
3. Сайт Algolist. Сортировка пузырьком – <http://algolist.manual.ru/sort/bubble_sort.php>.

4. Сайт Algolist. Сортировка вставками - <http://algolist.manual.ru/sort/insert_sort.php>.

5. Сайт Algolist. Сортировка Хоара - <http://algolist.manual.ru/sort/shell_sort.php>.

# Приложение

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <io.h>

#include <time.h>

#include <string.h>

#include <locale.h>

void Bub(unsigned long \*a, char b[10000][32], char \*direct, long number)

{

int i, j, puz;

char buffer[260];

for (i = 0; i < number; i++) //сортировка пузырьком

for (j = (number - 1); j > i; j--)

if (a[j] > a[j - 1])

{

puz = a[j];

a[j] = a[j - 1];

a[j - 1] = puz;

strncpy(buffer, b[j], 32);

strncpy(b[j], b[j - 1], 32);

strncpy(b[j - 1], buffer, 32);

}

}

void Cho(unsigned long \*a, char b[10000][32], char \*direct, long number)

{

int i, j, puz, vb;

char buffer[260];

for (i = 0; i < number; i++)//выбором

{

puz = i;

vb = a[i];

for (j = i + 1; j < number; j++)

if (a[j] > vb)

{

puz = j;

vb = a[j];

}

a[puz] = a[i];

a[i] = vb;

strncpy(buffer, b[puz], 32);

strncpy(b[puz], b[i], 32);

strncpy(b[i], buffer, 32);

}

}

void Sort(unsigned long \*a, char b[10000][32], char \*direct, long number) //Хоара

{

long i = 0, j = number - 1;

unsigned long temp, mid;

char buffer[260];

mid = a[number >> 1];

do

{

while (a[i] < mid)

i++;

while (a[j] > mid)

j--;

if (i <= j)

{

temp = a[i];

a[i] = a[j];

a[j] = temp;

strncpy(buffer, b[i], 32);

strncpy(b[i], b[j], 32);

strncpy(b[j], buffer, 32);

i++;

j--;

}

} while (i <= j);

if (j > 0) Sort(a, b, direct, j + 1);

if (number > i) Sort(a + i, b + i, direct, number - i);

}

void Ins(unsigned long \*a, char b[10000][32], char \*direct, long number) //Вставками

{

int x;

long i, j;

char buffer[260];

for (i = 0; i < number; i++)

{

x = a[i];

for (j = i - 1; j >= 0 && a[j] > x; j--)

{

a[j + 1] = a[j];

strncpy(buffer, b[j + 1], 32);

strncpy(b[j + 1], b[j], 32);

strncpy(b[j], buffer, 32);

}

a[j + 1] = x;

}

}

char file\_names[10000][32];

unsigned long file\_sizes[10000];

int main(void)

{

setlocale(LC\_ALL, "Rus");

int i;

struct \_finddata\_t c\_file;

intptr\_t hFile;

char direct[260];

int number = 0, sorting = 1;

clock\_t t1, t2;

double tt = 0;

printf("Ведите путь до директории\n");

do

{

gets\_s(direct, 260);

strcat(direct, "\*");

if ((hFile = \_findfirst(direct, &c\_file)) == -1L)

printf("Пути %s не существует! Попробуйте ещё раз:\n", direct);

else

{

printf("Список файлов в %s\n\n", direct);

printf("Файлы Размер\n", ' ');

printf("---- ----\n", ' ');

do

{

strncpy(file\_names[number], c\_file.name, 32);

file\_sizes[number] = c\_file.size;

printf("%-12.12s %10u\n", file\_names[number], file\_sizes[number]);

number++;

} while (\_findnext(hFile, &c\_file) == 0);

\_findclose(hFile);

printf("\nКоличество файлов: %d\n", number);

printf("\n");

}

} while ((hFile = \_findfirst(direct, &c\_file)) == -1L);

while (sorting != 0)

{

printf("Выберите сортировку:\n 1.Пузырьком\n 2.Выбором \n 3.Хоара \n 4.Вставками\n Если хотите выйти, введите 0\n");

scanf("%i", &sorting);

while ((sorting > 4) || (sorting < 0))

{

printf("Неверно выбрана сортировка!\n Выберите сортировку:\n 1.Пузырьком\n 2.Выбором \n 3.Хоара \n 4.Вставками\n Если хотите выйти, введите 0\n");

scanf("%i", &sorting);

}

switch (sorting)

{

case 1:

t1 = clock();

Bub(file\_sizes, file\_names, direct, number);

t2 = clock();

break;

case 2:

t1 = clock();

Cho(file\_sizes, file\_names, direct, number);

t2 = clock();

break;

case 3:

{

t1 = clock();

Sort(file\_sizes, file\_names, direct, number);

t2 = clock();

break;

}

case 4:

{

t1 = clock();

Ins(file\_sizes, file\_names, direct, number);

t2 = clock();

break;

}

case 0:

sorting = 0;

break;

}

tt = double(t2 - t1) / CLOCKS\_PER\_SEC;

//вывод результата

printf("Файлы Размер\n", ' ');

printf("---- ----\n", ' ');

if ((sorting == 1) || (sorting == 2))

{

for (i = 0; i < number; i++)

printf("%-12.12s %10u\n", file\_names[i], file\_sizes[i]);

}

else

for (i = number - 1; i >= 0; i--)

printf("%-12.12s %10u\n", file\_names[i], file\_sizes[i]);

printf("\nВремя работы сортировки :%f\n\n", tt);

}

}