Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

кафедра «Вычислительных методов и программирования»

**ОТЧЁТ**

По лабораторным работам №5 и №6

«СОРТИРОВКА ПО КЛЮЧУ ОДНОМЕРНЫХ МАССИВОВ СТРУКТУР. АЛГОРИТМЫ ПОИСКА»

Выполнил:

Студент ФИТиУ

гр. 020601

Гудков А. С.

Вариант №7

Проверил:

ассистент кафедры ВМиП

Беспалов С. А.

Минск 2021

***Цель работы:***

Научиться сортировать по ключу одномерные массивы структур различными способами, а также при помощи алгоритмов поиска находить определённые элементы по заданному ключу. Написать и отладить программу для сортировки и поиска.

***Индивидуальное задание (лаб. №5):***

7. Упорядочить по неубыванию массив структур по заданному ключу. Ключ: код детали. Методы сортировки: QuickSort и сортировка выбором.

***Индивидуальное задание (лаб. №6):***

7. Найти в отсортированном массиве структур деталь с заданным кодом. Методы поиска: полный перебор и двоичный.

***Текст программы:***

#include <iostream>

#include <stdio.h>

#include <io.h>

using namespace std;

int **menu**(); // MENU

void **fl\_add**(); // Enter data to file

void **fl\_read**(); // Read data from file

void **sort\_con**(); // Sort details in console

void **sort\_fl**(); // Sort details in file

void **sort**(); // Sort MENU

void **search\_con**(); // Search details in console

void **search\_fl**(); // Search details in file

int **search**(); // Search MENU

void **sort\_quick**(); // Quick Sort

void **sort\_selection**(); // Selection Sort

int **search\_linear**(int); // Linear Search

int **search\_binary**(int); // Binary Search

FILE\* fl;

struct dinfo

{

int code;

int amount;

int month;

} detail, \* DETAILS;

int n = 0;

char flname[30];

bool no\_fl\_name = true;

int **main**()

{

while (true)

{

switch (menu())

{

case 1: **fl\_add**(); break;

case 2: **fl\_read**(); break;

case 3: **sort\_con**(); break;

case 4: **sort\_fl**(); break;

case 5: **search\_con**(); break;

case 6: **search\_fl**(); break;

case 7: return 0; break;

default: cout << "Enter correct!" << endl;

}

cout << endl;

system("pause");

system("cls");

}

}

int **menu**()

{

cout << "+------------------------------+" << endl;

cout << "| 1. Enter data to file |" << endl;

cout << "| 2. Read data from file |" << endl;

cout << "| 3. Sort details in console |" << endl;

cout << "| 4. Sort details in file |" << endl;

cout << "| 5. Search details in console |" << endl;

cout << "| 6. Search details in file |" << endl;

cout << "| 7. Exit |" << endl;

cout << "+------------------------------+" << endl;

int k;

cin >> k;

cout << endl;

return k;

}

char\* **fl\_name\_get**()

{

if (no\_fl\_name)

{

cout << "Enter file name: "; cin >> flname;

no\_fl\_name = false;

}

return flname;

}

void **fl\_add**()

{

fopen\_s(&fl, fl\_name\_get(), "ab");

if (fl == NULL)

{

fopen\_s(&fl, fl\_name\_get(), "wb");

if (fl == NULL)

{

cout << "FILE CREATE ERROR" << endl;

return;

}

}

char chr;

do

{

cout << "Enter detail code: "; cin >> detail.code;

cout << "Enter detail amount: "; cin >> detail.amount;

cout << "Enter detail month: "; cin >> detail.month;

fwrite(&detail, sizeof(dinfo), 1, fl);

cout << endl << "Enter more? (y/n) "; cin >> chr;

} while (chr == 'y');

fclose(fl);

}

void **fl\_read**()

{

fopen\_s(&fl, fl\_name\_get(), "rb");

if (fl == NULL)

{

cout << "FILE READ ERROR" << endl;

return;

}

n = \_filelength(\_fileno(fl)) / sizeof(dinfo);

DETAILS = new dinfo[n];

fread(DETAILS, sizeof(dinfo), n, fl);

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

{

cout << "Code - " << DETAILS[i].code << " Amount - " << DETAILS[i].amount

<< " Month - " << DETAILS[i].month << endl;

}

fclose(fl);

}

*//-+-+-+-+-+-+-+-+-+-+-SORT-+-+-+-+-+-+-+-+-+-+-*

void **sort\_con**()

{

fopen\_s(&fl, fl\_name\_get(), "rb");

if (fl == NULL)

{

cout << "FILE READ ERROR" << endl;

return;

}

n = \_filelength(\_fileno(fl)) / sizeof(dinfo);

DETAILS = new dinfo[n];

fread(DETAILS, sizeof(dinfo), n, fl);

sort();

cout << "Sorted!" << endl;

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

{

cout << "Code - " << DETAILS[i].code << " Amount - " <<

DETAILS[i].amount << " Month - " << DETAILS[i].month << endl;

}

fclose(fl);

}

void **sort\_fl**()

{

fopen\_s(&fl, fl\_name\_get(), "rb");

if (fl == NULL)

{

cout << "FILE OPEN ERROR" << endl;

return;

}

char flname2[20];

cout << "Enter result file name: "; cin >> flname2;

FILE\* fl2;

fopen\_s(&fl2, flname2, "wt");

if (fl2 == NULL)

{

cout << "FILE CREATE ERROR" << endl;

return;

}

n = \_filelength(\_fileno(fl)) / sizeof(dinfo);

DETAILS = new dinfo[n];

fread(DETAILS, sizeof(dinfo), n, fl);

*sort*();

fprintf(fl2, "%s\n", "Sort key - Code");

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

{

fprintf(fl2, "%s%d %s%d %s%d\n", "Code - ", DETAILS[i].code,

"Amount - ", DETAILS[i].amount, "Month - ", DETAILS[i].month);

}

cout << "Done" << endl;

\_fcloseall();

}

void **sort**()

{

cout << "1 - Quick Sort" << endl;

cout << "2 - Selection Sort" << endl;

cout << "------------------" << endl;

int k;

while (true)

{

cin >> k;

switch (k)

{

case 1: **sort\_quick**(); return;

case 2: **sort\_selection**(); return;

default: cout << "Enter correct!" << endl;

}

}

}

void **sort\_quick**()

{

struct St

{

int L;

int R;

int k;

} stack[10];

int i, j, left, right, k;

dinfo temp, x;

k = 0;

stack[k].L = 0;

stack[k].R = n-1;

while (k != -1)

{

left = stack[k].L;

right = stack[k].R;

k--;

while (left < right)

{

i = left;

j = right;

x = DETAILS[(i + j) / 2];

while (i <= j)

{

while (DETAILS[i].code < x.code) i++;

while (DETAILS[j].code > x.code) j--;

if (i <= j)

{

temp = DETAILS[i];

DETAILS[i] = DETAILS[j];

DETAILS[j] = temp;

i++;

j--;

}

}

if ((j-left) < (right-i))

{

if (i < right)

{

k++;

stack[k].L = i;

stack[k].R = right;

}

right = j;

}

else

{

if (left < j)

{

k++;

stack[k].L = left;

stack[k].R = j;

}

left = i;

}

}

}

}

void **sort\_selection**()

{

int i\_temp;

dinfo temp;

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

{

i\_temp = i;

for (int j = i+1; j < n; j++)

if (DETAILS[i\_temp].code > DETAILS[j].code)

i\_temp = j;

if (i\_temp != i)

{

temp = DETAILS[i\_temp];

DETAILS[i\_temp] = DETAILS[i];

DETAILS[i] = temp;

}

}

}

*//-+-+-+-+-+-+-+-+-+-+-SEARCH-+-+-+-+-+-+-+-+-+-+-*

void **search\_con**()

{

fopen\_s(&fl, fl\_name\_get(), "rb");

if (fl == NULL)

{

cout << "FILE READ ERROR" << endl;

return;

}

n = \_filelength(\_fileno(fl)) / sizeof(dinfo);

DETAILS = new dinfo[n];

fread(DETAILS, sizeof(dinfo), n, fl);

int i = **search**();

if (i != -1)

cout << "Code - " << DETAILS[i].code << " Amount - " <<

DETAILS[i].amount << " Month - " << DETAILS[i].month << endl;

else

cout << "Detail not found!" << endl;

fclose(fl);

}

void **search\_fl**()

{

fopen\_s(&fl, fl\_name\_get(), "rb");

if (fl == NULL)

{

cout << "FILE OPEN ERROR" << endl;

return;

}

char flname2[20];

cout << "Enter result file name: "; cin >> flname2;

FILE\* fl2;

fopen\_s(&fl2, flname2, "wt");

if (fl2 == NULL)

{

cout << "FILE CREATE ERROR" << endl;

return;

}

n = \_filelength(\_fileno(fl)) / sizeof(dinfo);

DETAILS = new dinfo[n];

fread(DETAILS, sizeof(dinfo), n, fl);

int i = **search**();

if (i != -1)

fprintf(fl2, "%s%d %s%d %s%d\n", "Code - ", DETAILS[i].code,

"Amount - ", DETAILS[i].amount, "Month - ", DETAILS[i].month);

else

fprintf(fl2, "%s", "Detail no found!");

cout << "Done" << endl;

\_fcloseall();

}

int **search**()

{

**sort\_quick**();

int x;

cout << "Enter detail code: "; cin >> x;

cout << "1 - Linear Search" << endl;

cout << "2 - Binary Search" << endl;

cout << "-----------------" << endl;

int k;

while (true)

{

cin >> k;

switch (k)

{

case 1: return **search\_linear**(x);

case 2: return **search\_binary**(x);

default: cout << "Enter correct!" << endl;

}

}

}

int **search\_linear**(int x)

{

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

if (DETAILS[i].code == x) return i;

return -1;

}

int **search\_binary**(int x)

{

int i = 0, j = n-1, m;

while (i < j)

{

m = (i + j) / 2;

if (x > DETAILS[m].code) i = m+1;

else j = m;

}

if (DETAILS[i].code == x) return i;

else return -1;

}

***Результат работы программы:***

 

 