НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Лабораторна робота №3

З дисципліни: «Об’єктно-орієнтоване програмування»

Виконав:

Студент групи КВ-51

Тимошенко Ігор

Перевірив

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КИЇВ 2016

scanner.h

#pragma once

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

typedef struct {

char manufacturer[127];

int year;

char model[128];

float price;

int x\_size;

int y\_size;

int optr;

} SCAN\_INFO;

typedef struct {

int rec\_nmb; // number of records

SCAN\_INFO \*recs; // records

} RECORD\_SET;

int write\_scan(FILE \*dba, const SCAN\_INFO \*scanner); // 1

int write\_from\_csv(const char \*dba\_path, const char \*csv\_path); // 2

// functions for qsort()

int manufacturer\_cmp(const void \*a1, const void \*a2);

int year\_cmp(const void \*a1, const void \*a2);

int model\_cmp(const void \*a1, const void \*a2);

int price\_cmp(const void \*a1, const void \*a2);

int x\_size\_cmp(const void \*a1, const void \*a2);

int y\_size\_cmp(const void \*a1, const void \*a2);

int optr\_cmp(const void \*a1, const void \*a2);

int make\_index(char \*dba\_path, char \*field\_name); // 3

RECORD\_SET \*get\_recs\_by\_index(char \*dba\_path, char \*indx\_field); // 4

void reindex(char \*dba\_path); // 5

int del\_scan(char \*dba\_path, int n); // 6

int print\_to\_txt(char \*dba\_path, char \*txt\_path, float price); // 7

scanner.c

#include "scanner.h"

// 1

int write\_scan(FILE \*dba, const SCAN\_INFO \*scanner)

{

SCAN\_INFO \*buff;

int num = 0;

if ((buff = (SCAN\_INFO\*)malloc(sizeof(SCAN\_INFO))) == NULL)

return 1;

fseek(dba, 0, SEEK\_SET);

if (!feof(dba))

{

fread(&num, sizeof(int), 1, dba);

while (!feof(dba))

{

fread(buff, sizeof(SCAN\_INFO), 1, dba);

if (memcmp(scanner, buff, sizeof(SCAN\_INFO)) == 0)

{

free(buff);

return 0;

}

}

free(buff);

fseek(dba, 0, SEEK\_SET);

}

num++;

fwrite(&num, sizeof(int), 1, dba);

fseek(dba, 0, SEEK\_END);

fwrite(scanner, sizeof(SCAN\_INFO), 1, dba);

return 0;

}

// 2

int write\_from\_csv(const char \*dba\_path, const char \*csv\_path)

{

FILE \*dba;

FILE \*csv;

SCAN\_INFO \*recs;

if ((dba = fopen(dba\_path, "w+")) == NULL)

{

return 1;

}

if ((csv = fopen(csv\_path, "r")) == NULL)

{

fclose(dba);

return 1;

}

if ((recs = (SCAN\_INFO\*)malloc(sizeof(SCAN\_INFO))) == NULL)

{

fclose(dba);

fclose(csv);

return 1;

}

putw(0,dba);

while (!feof(csv))

{

fscanf(csv, "%[A-z0-9];%[A-z0-9];%d;%f;%d;%d;%d\n", recs->manufacturer, recs->model, &(recs->year), &(recs->price), &(recs->x\_size), &(recs->y\_size), &(recs->optr));

if (write\_scan(dba, recs))

break;

}

free(recs);

fclose(dba);

fclose(csv);

return 0;

}

// 3

int make\_index(char \*dba\_path, char \*field\_name)

{

FILE \*idx, \*dba;

SCAN\_INFO \*\*recs, \*buff;

char \*idx\_path;

int i, j, num = 0;

int len = strlen(field\_name),

fldlen = strlen("index/"); //folder len

if ((dba = fopen(dba\_path, "rb")) == NULL)

return 1;

if ((idx\_path = (char\*)malloc((len + 4 + fldlen)\*sizeof(char))) == NULL)

{

fclose(dba);

return 1;

}

strcpy(idx\_path, "index/");

strcpy(idx\_path + fldlen, field\_name);

strcpy(idx\_path + fldlen + len, ".idx");

if ((idx = fopen(idx\_path, "w")) == NULL)

{

free(idx\_path);

fclose(dba);

return 1;

}

free(idx\_path);

fread(&num, sizeof(int), 1, dba);

if (num > 0)

{

recs = (SCAN\_INFO\*\*)malloc(num \* sizeof(SCAN\_INFO\*));

buff = (SCAN\_INFO\*)malloc(sizeof(SCAN\_INFO));

if (recs == NULL || buff == NULL)

{

fclose(dba);

fclose(idx);

return 1;

}

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

{

if ((recs[i] = (SCAN\_INFO\*)malloc(sizeof(SCAN\_INFO))) == NULL)

{

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

free(recs[j]);

free(buff);

free(recs);

fclose(dba);

fclose(idx);

return 1;

}

fread(recs[i], sizeof(SCAN\_INFO), 1, dba);

}

}

if (field\_name == "manufacturer")

qsort(recs, num, sizeof(SCAN\_INFO\*), manufacturer\_cmp);

else if (field\_name == "year")

qsort(recs, num, sizeof(SCAN\_INFO\*), year\_cmp);

else if (field\_name == "model")

qsort(recs, num, sizeof(SCAN\_INFO\*), model\_cmp);

else if (field\_name == "price")

qsort(recs, num, sizeof(SCAN\_INFO\*), price\_cmp);

else if (field\_name == "x\_size")

qsort(recs, num, sizeof(SCAN\_INFO\*), x\_size\_cmp);

else if (field\_name == "y\_size")

qsort(recs, num, sizeof(SCAN\_INFO\*), y\_size\_cmp);

else if (field\_name == "optr")

qsort(recs, num, sizeof(SCAN\_INFO\*), optr\_cmp);

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

{

fseek(dba, sizeof(int), SEEK\_SET);

for (j = 0; j < num; j++)

{

fread(buff, sizeof(SCAN\_INFO), 1, dba);

if (recs[i] != NULL && strcmp(buff->manufacturer, recs[i]->manufacturer) == 0 && strcmp(buff->model, recs[i]->model) == 0)

{

fprintf(idx, "%d ", j);

free(recs[i]);

recs[i] = NULL;

}

}

}

free(recs);

free(buff);

fclose(dba);

fclose(idx);

return 0;

}

// functions for qsort()

int manufacturer\_cmp(const void \*a1, const void \*a2)

{

return strcmp((\*(SCAN\_INFO\*\*)a1)->manufacturer, (\*(SCAN\_INFO\*\*)a2)->manufacturer);

}

int year\_cmp(const void \*a1, const void \*a2)

{

return (\*(SCAN\_INFO\*\*)a1)->year - (\*(SCAN\_INFO\*\*)a2)->year;

}

int model\_cmp(const void \*a1, const void \*a2)

{

return strcmp((\*(SCAN\_INFO\*\*)a1)->model, (\*(SCAN\_INFO\*\*)a2)->model);

}

int price\_cmp(const void \*a1, const void \*a2)

{

return (int)((\*(SCAN\_INFO\*\*)a1)->price - (\*(SCAN\_INFO\*\*)a2)->price);

}

int x\_size\_cmp(const void \*a1, const void \*a2)

{

return (\*(SCAN\_INFO\*\*)a1)->x\_size - (\*(SCAN\_INFO\*\*)a2)->x\_size;

}

int y\_size\_cmp(const void \*a1, const void \*a2)

{

return (\*(SCAN\_INFO\*\*)a1)->y\_size - (\*(SCAN\_INFO\*\*)a2)->y\_size;

}

int optr\_cmp(const void \*a1, const void \*a2)

{

return (\*(SCAN\_INFO\*\*)a1)->optr - (\*(SCAN\_INFO\*\*)a2)->optr;

}

// 4

RECORD\_SET \*get\_recs\_by\_index(char \*dba\_path, char \*indx\_field)

{

RECORD\_SET \*set;

FILE \*dba, \*idx;

int i = 0, j = 0;

if ((dba = fopen(dba\_path, "rb")) == NULL)

return NULL;

if ((idx = fopen(indx\_field, "rb")) == NULL)

{

fclose(dba);

return NULL;

}

if ((set = (RECORD\_SET\*)malloc(sizeof(RECORD\_SET))) == NULL)

{

fclose(dba);

fclose(idx);

return NULL;

}

fread(&(set->rec\_nmb), sizeof(int), 1, dba);

if (set->rec\_nmb > 0)

{

if ((set->recs = (SCAN\_INFO\*)malloc(set->rec\_nmb \* sizeof(SCAN\_INFO))) == NULL)

{

free(set);

fclose(dba);

fclose(idx);

return NULL;

}

for (i = 0; i < set->rec\_nmb; i++)

{

fscanf(idx, "%d ", &j);

fseek(dba, sizeof(int), SEEK\_SET);

fseek(dba, j \* sizeof(SCAN\_INFO), SEEK\_CUR);

fread(set->recs + i, sizeof(SCAN\_INFO), 1, dba);

}

}

fclose(dba);

fclose(idx);

return set;

}

//5

void reindex(char \*dba\_path)

{

make\_index(dba\_path, "manufacturer");

make\_index(dba\_path, "year");

make\_index(dba\_path, "model");

make\_index(dba\_path, "price");

make\_index(dba\_path, "x\_size");

make\_index(dba\_path, "y\_size");

make\_index(dba\_path, "optr");

}

//6

int del\_scan(char \*dba\_path, int n)

{

FILE \*dba;

int num, i, j = 0;

SCAN\_INFO \*recs;

if ((dba = fopen(dba\_path, "rb")) == NULL)

return 1;

fread(&num, sizeof(int), 1, dba);

if (num == 0 || n >= num)

{

fclose(dba);

return 1;

}

if ((recs = (SCAN\_INFO\*)malloc(num \* sizeof(SCAN\_INFO))) == NULL)

{

fclose(dba);

return 1;

}

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

{

fread(recs + j, sizeof(SCAN\_INFO), 1, dba);

if (i == n) continue;

j++;

}

if ((dba = freopen(dba\_path, "wb", dba)) == NULL)

{

free(recs);

fclose(dba);

return 1;

}

rewind(dba);

num--;

fwrite(&num, sizeof(int), 1, dba);

fwrite(recs, sizeof(SCAN\_INFO), num, dba);

free(recs);

fclose(dba);

reindex(dba\_path);

return 0;

}

//7

int print\_to\_txt(char \*dba\_path, char \*txt\_path, float price)

{

FILE \*dba;

FILE \*txt;

SCAN\_INFO \*recs;

int num = 0, i = 0;

if ((dba = fopen(dba\_path, "r")) == NULL)

return 1;

if ((txt = fopen(txt\_path, "w")) == NULL)

{

fclose(dba);

return 1;

}

fread(&num, sizeof(int), 1, dba);

if (num > 0)

{

if ((recs = (SCAN\_INFO\*)malloc(num \* sizeof(SCAN\_INFO))) == NULL)

{

fclose(dba);

fclose(txt);

return 1;

}

fread(recs, sizeof(SCAN\_INFO), num, dba);

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

{

if (recs[i].price <= price)

fprintf(txt, "%s;%s;%d;%g;%d;%d;%d\n", recs[i].manufacturer, recs[i].model, recs[i].year, recs[i].price, recs[i].x\_size, recs[i].y\_size, recs[i].optr);

}

free(recs);

}

fclose(dba);

fclose(txt);

return 0;

}

scanner\_db.c

#include "scanner.h"

int main()

{

if (write\_from\_csv("Scanners.dba", "Scanners.csv"))

printf("Error with .dba creating\n");

reindex("Scanners.dba");

if (get\_recs\_by\_index("Scanners.dba", "index/manufacturer.idx") == NULL)

printf("Error in get\_recs\_by\_index()\n");

if (del\_scan("Scanners.dba", 5))

printf("Error with deleting scanner\n");

if (print\_to\_txt("Scanners.dba", "Scanners\_by\_price.txt", 1000.0))

printf("Error with input to txt\n");

return 0;

}