

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

# Кафедра системного програмування і спеціалізованих комп’ютерних систем

**Лабораторна робота №1**

з дисципліни

**«Об'єктно Орієнтоване Програмування»**

Тема: ***«****Файловый ввід-вивід. База даних****»***

Виконав: студент ІI курсу

ФПМ групи КВ-61

Бідяк М. А.

Перевірив:

Київ-2017

**Завдання**

Дані про планшетні сканери зберігаються в структурі виду

typedef struct {

int id;//унікальний ідентифікатор

char\* manufacturer;// виготовлювач

int year; // рік виготовлення

char \*model;// найменування моделі

float price;// ціна

int x\_size;// горизонтальний розмір області сканування

int y\_size;// вертикальний розмір області сканування

} SCAN\_INFO;

1. Написати функцію, *create\_db(const cha\* csv, const char \*db)* що записує з текстового файлу csv. в бінарний файл *db* дані про сканери наведеної структури. Формат файлу csv(Comma Separated Values) передбвчає, що кожному рядку файлу зберігається один запис, а його поля відокремлюються друг від друга символом “;”.

Структура файлу *db*: у перших sizeof( int ) байтах розміщається значення типу int, що визначає кількість записів у файлі, далі без пропусків розміщаються записи про сканери. До кожного рядка з csv додається поле *id* типу *int*, яке є унікальнім ідентифікатором запису. Функція повинна стежити, щоб у файлі не було повторюваних записів.

1. Написати функцію

//returns 1 on success and 0 otherwise

int make\_index(const char \*db, const char \*field\_name),

що створює індексний файл для заданого db-файлу. Індексний файл містить номера записів db-файлу, відсортовані в порядку неубування по field\_name. Ім'я індексного файлу складається з field\_name і розширення *idx*. Наприклад, db-файл містить наступні записи

1. Написати функцію RECORD\_SET \* get\_recs\_by\_index(const char \*dba , char \*indx\_field ), що повертає покажчик RECORD\_SET \* на набір записів, отриманий за допомогою індексного файлу indx\_field.idx. Набір записів визначається наступною структурою

typedef struct{

int rec\_nmb;//number of records

SCAN\_INFO \*recs;//records

}RECORD\_SET;

4. Написати функцію void reindex(const char \*db), що створює індексні файли по ВСІХ полях структури SCAN\_INFO.

5. Написати функцію void del\_scanner(const char \*db, int id) , що видаляє з db-файлу запис з заданим id. Після видалення запису необхідно викликати функцію reindex().

6. Написати функцію void add\_scanner(const char \*db, const char\* scanner\_str) , що додає до db-файлу запис з *c*s*v* рядка scanner\_str. При цьому треба забезпечити унікальність поля id. Потім необхідно викликати функцію reindex().

7.Написати функцію void print\_db(const char \*db), що читає дані про сканери з db-файлу й виводить їх у файл \*.txt.

8.Написати функцію RECORD\_SET\* select(const char \*db, const char \*field, const char \*value), що повертає покажчик RECORD\_SET \* на набір записів, в яких значення поля field співпадає з value. Наприклад,

RECORD\_SET\* rs = select("Scanners.db", "y\_size", "297");

9.Написати функцію void print\_rec\_set(RECORD\_SET \*rs), яка друкує вміст rs.

**Код програми**

Scanner.h

#ifndef SCANNER\_H

#define SCANNER\_H

#include <io.h> //\_chsize

#include <stdlib.h> //atof and etc.

#include <conio.h> //\_getch

#include <stdio.h> //standart fucntion and etc.

#include <string.h> //strstr

#pragma warning( disable : 4244 34 )

typedef struct {

int id;//унікальний ідентифікатор

char manufacturer[128];// виготовлювач

int year; // рік виготовлення

char model[128];// найменування моделі

float price;// ціна

int x\_size;// горизонтальний розмір області сканування

int y\_size;// вертикальний розмір області сканування

} SCAN\_INFO;

typedef struct {

int rec\_nmb; //number of records

SCAN\_INFO \*recs; //records

}RECORD\_SET;

void create\_db(const char\* csv, const char \*db);

void print\_db(const char\*db);

int make\_index(const char \*db, const char \*field\_name);

void readIDX(const char \*field\_name);

int manufacturer(const void\*x1, const void\*x2);

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

void reindex(const char \*db);

int year\_cmp(const void \*val1, const void \*val2);

int model\_cmp(const void \*val1, const void \*val2);

int price\_cmp(const void \*val1, const void \*val2);

int x\_size\_cmp(const void \*val1, const void \*val2);

int y\_size\_cmp(const void \*val1, const void \*val2);

void print\_rec\_set(RECORD\_SET \*rs);

RECORD\_SET\* select(const char \*db, const char \*field, const char \*value);

void add\_scanner(const char \*db, const char\* scanner\_str);

void del\_scanner(const char \*db, int id);

#endif /\* SCANNER\_H \*/

Scanner.c

#include "Scanner.h"

void create\_db(const char\* csv, const char \*db)

{

FILE \*CSV, \*DB, \*TrueDB;

SCAN\_INFO \*csv\_point, \*db\_point;

unsigned int copy\_flag = 0, numberOfStructs = 0;

csv\_point = (SCAN\_INFO\*)malloc(sizeof(SCAN\_INFO));

db\_point = (SCAN\_INFO\*)malloc(sizeof(SCAN\_INFO));

if (csv\_point == NULL) return;

if (db\_point == NULL) return;

CSV = fopen(csv, "r");

if (CSV == NULL) return;

DB = fopen("tmp.bin", "w+");

if (DB == NULL) return;

while (!feof(CSV)) {

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

printf("%s; %s; %d; %.2f; %d; %d \n", (csv\_point->manufacturer), (csv\_point->model), (csv\_point->year), (csv\_point->price), (csv\_point->x\_size), (csv\_point->y\_size));

if (!feof(DB)) {

rewind(DB);

while (!feof(DB))

{

fread(db\_point, sizeof(SCAN\_INFO), 1, DB);

if ((strstr(csv\_point->manufacturer, db\_point->manufacturer) != 0) && (strstr(csv\_point->model, db\_point->model) != 0) && (csv\_point->year == db\_point->year) && (csv\_point->price == db\_point->price) && (csv\_point->x\_size == db\_point->x\_size) && (csv\_point->y\_size == db\_point->y\_size))

{

copy\_flag = 1;

break;

}

else copy\_flag = 0;

}

if (copy\_flag == 0)

{

fseek(DB, 0, SEEK\_END);

fwrite(csv\_point, (sizeof(SCAN\_INFO)), 1, DB);

numberOfStructs++;

copy\_flag = 1;

}

}

else

{

fwrite(csv\_point, sizeof(SCAN\_INFO), 1, DB);

numberOfStructs++;

}

}

fclose(DB);

fclose(CSV);

free(csv\_point);

DB = fopen("tmp.bin", "r");

if (DB == NULL) return;

TrueDB = fopen(db, "w");

if (TrueDB == NULL) return;

fwrite(&numberOfStructs, sizeof(int), 1, TrueDB);

while (fread(db\_point, sizeof(SCAN\_INFO), 1, DB))

{

fwrite(db\_point, sizeof(SCAN\_INFO), 1, TrueDB);

}

free(db\_point);

fclose(DB);

fclose(TrueDB);

remove("tmp.bin");

}

void print\_db(const char\* db)

{

FILE \*DB, \*TXT;

SCAN\_INFO \*db\_point;

int numbersOfStructs = 0;

db\_point = (SCAN\_INFO\*)malloc(sizeof(SCAN\_INFO));

if (db\_point == NULL) return ;

DB = fopen(db, "r");

if (DB == NULL) return ;

TXT = fopen("Scanners.txt", "w");

if (TXT == NULL) return ;

fread(&numbersOfStructs, sizeof(int), 1, DB);

if (numbersOfStructs == 0) return;

printf("\tNumber of Structs = %d\n", numbersOfStructs);

while (fread(db\_point, sizeof(SCAN\_INFO), 1, DB))

{

fprintf(TXT, "%s; %s; %d; %.2f; %d; %d \n",(db\_point->manufacturer), (db\_point->model), (db\_point->year), (db\_point->price), (db\_point->x\_size), (db\_point->y\_size));

printf("%s; %s; %d; %.2f; %d; %d \n", (db\_point->manufacturer), (db\_point->model), (db\_point->year), (db\_point->price), (db\_point->x\_size), (db\_point->y\_size));

}

fclose(DB);

free(db\_point);

}

int make\_index(const char \*db, const char \*field\_name)

{

FILE \*DB, \*IDX;

SCAN\_INFO \*db\_point, \*vect;

short unsigned int j = 0;

int numbersOfStructs = 0;

db\_point = (SCAN\_INFO\*)malloc(sizeof(SCAN\_INFO));

if (db\_point == NULL) return -1;

DB = fopen(db, "r");

if (DB == NULL)

{

free(db\_point);

return -1;

}

IDX = fopen(field\_name, "w+");

if (IDX == NULL)

{

free(db\_point);

fclose(DB);

return -1;

}

fread(&numbersOfStructs,sizeof(int),1,DB);

vect = (SCAN\_INFO\*)malloc(sizeof(SCAN\_INFO)\*numbersOfStructs);

while (j < numbersOfStructs)

{

fread(&(vect[j]), sizeof(SCAN\_INFO), 1, DB);

(vect[j].id) = j;

j++;

}

if ((strcmp(field\_name, "manufacturer.idx")) == 0)

qsort(vect, numbersOfStructs, sizeof(SCAN\_INFO), manufacturer);

else if ((strcmp(field\_name, "year.idx")) == 0)

qsort(vect, numbersOfStructs, sizeof(SCAN\_INFO), year\_cmp);

else if ((strcmp(field\_name, "model.idx")) == 0)

qsort(vect, numbersOfStructs, sizeof(SCAN\_INFO), model\_cmp);

else if ((strcmp(field\_name, "price.idx")) == 0)

qsort(vect, numbersOfStructs, sizeof(SCAN\_INFO), price\_cmp);

else if ((strcmp(field\_name, "x\_size.idx")) == 0)

qsort(vect, numbersOfStructs, sizeof(SCAN\_INFO), x\_size\_cmp);

else if ((strcmp(field\_name, "y\_size.idx")) == 0)

qsort(vect, numbersOfStructs, sizeof(SCAN\_INFO), y\_size\_cmp);

j = 0;

while (j < numbersOfStructs)

{

fwrite(&(vect[j]), sizeof(int), 1, IDX);

j++;

}

free(vect);

fclose(DB);

fclose(IDX);

free(db\_point);

return 1;

}

void readIDX(const char \*field\_name)

{

FILE \*IDX;

int x = 0;

IDX = fopen(field\_name, "rb");

if (IDX == NULL) return;

while (fread(&x, sizeof(int), 1, IDX)) {

printf(" %d ", x);

}

printf("\n\n");

fclose(IDX);

}

int manufacturer (const void \*x1, const void \*x2)

{

return strcmp(((SCAN\_INFO\*)x1)->manufacturer, ((SCAN\_INFO\*)x2)->manufacturer) >= 0;

//&& strcmp(((SCAN\_INFO\*)x1)->model, ((SCAN\_INFO\*)x2)->model) >= 0;

}

int year\_cmp(const void \*val1, const void \*val2)

{

return ((SCAN\_INFO\*)val1)->year - ((SCAN\_INFO\*)val2)->year;

}

int model\_cmp(const void \*val1, const void \*val2)

{

return strcmp(((SCAN\_INFO\*)val1)->model, ((SCAN\_INFO\*)val2)->model);

}

int price\_cmp(const void \*val1, const void \*val2)

{

return ((SCAN\_INFO\*)val1)->price - ((SCAN\_INFO\*)val2)->price;

}

int x\_size\_cmp(const void \*val1, const void \*val2)

{

return ((SCAN\_INFO\*)val1)->x\_size - ((SCAN\_INFO\*)val2)->x\_size;

}

int y\_size\_cmp(const void \*val1, const void \*val2)

{

return ((SCAN\_INFO\*)val1)->y\_size - ((SCAN\_INFO\*)val2)->y\_size;

}

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

{

FILE \*DB, \*IDX;

RECORD\_SET \*rec;

int i, fieldID;

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

{

return NULL;

}

if ((DB = fopen(dba, "rb")) == NULL)

{

fclose(IDX);

return NULL;

}

if ((rec = (RECORD\_SET\*)malloc(sizeof(int) + sizeof(SCAN\_INFO\*))) == NULL)

{

fclose(IDX);

fclose(DB);

return NULL;

}

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

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

{

free(rec);

fclose(IDX);

fclose(DB);

return NULL;

}

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

{

fread(&fieldID, sizeof(int), 1, IDX);

fseek(DB, fieldID \* sizeof(SCAN\_INFO) + sizeof(int), SEEK\_SET);

fread(&(rec->recs[i]), sizeof(SCAN\_INFO), 1, DB);

}

fclose(DB);

fclose(IDX);

return rec;

}

RECORD\_SET\* select(const char \*db, const char \*field, const char \*value)

{

FILE \*DB;

RECORD\_SET \*rec;

SCAN\_INFO \*db\_point;

int numbersOfStructs = 0, i = 0;

float val;

if ((DB = fopen(db, "rb")) == NULL)

{

return NULL;

}

if ((db\_point = (SCAN\_INFO\*)malloc(sizeof(SCAN\_INFO))) == NULL)

{

fclose(DB);

return NULL;

}

if ((rec = (RECORD\_SET\*)malloc(sizeof(int) + sizeof(SCAN\_INFO\*))) == NULL)

{

free(db\_point);

fclose(DB);

return NULL;

}

rec->rec\_nmb = 0;

fread(&numbersOfStructs, sizeof(int), 1, DB);

if ((strcmp(field, "manufacturer")) == 0)

{

while (fread(db\_point, sizeof(SCAN\_INFO), 1, DB))

if ((strcmp(db\_point->manufacturer, value)) == 0)

rec->rec\_nmb++;

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

{

free(db\_point);

free(rec);

fclose(DB);

return NULL;

}

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

while (fread(db\_point, sizeof(SCAN\_INFO), 1, DB))

if ((strcmp(db\_point->manufacturer, value)) == 0)

{

(rec->recs[i]) = \*db\_point;

i++;

}

if (i <= 0)

{

printf("Fuction \"Select\" can't find this value");

return NULL;

}

}

else if ((strcmp(field, "model")) == 0)

{

while (fread(db\_point, sizeof(SCAN\_INFO), 1, DB))

if ((strcmp(db\_point->model, value)) == 0)

rec->rec\_nmb++;

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

{

free(db\_point);

free(rec);

fclose(DB);

return NULL;

}

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

while (fread(db\_point, sizeof(SCAN\_INFO), 1, DB))

if ((strcmp(db\_point->model, value)) == 0)

{

(rec->recs[i]) = \*db\_point;

i++;

}

if (i <= 0)

{

printf("Fuction \"Select\" can't find this value");

return NULL;

}

}

else if ((strcmp(field, "year")) == 0)

{

while (fread(db\_point, sizeof(SCAN\_INFO), 1, DB))

if (db\_point->year == atoi(value))

rec->rec\_nmb++;

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

{

free(db\_point);

free(rec);

fclose(DB);

return NULL;

}

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

while (fread(db\_point, sizeof(SCAN\_INFO), 1, DB))

if (db\_point->year == atoi(value))

{

(rec->recs[i]) = \*db\_point;

i++;

}

if( i <= 0)

{

printf("Fuction \"Select\" can't find this value");

return NULL;

}

}

else if ((strcmp(field, "price")) == 0)

{

val = atof(value);

while (fread(db\_point, sizeof(SCAN\_INFO), 1, DB))

if (db\_point->price == val)

rec->rec\_nmb++;

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

{

free(db\_point);

free(rec);

fclose(DB);

return NULL;

}

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

while (fread(db\_point, sizeof(SCAN\_INFO), 1, DB))

if (db\_point->price == val)

{

(rec->recs[i]) = \*db\_point;

i++;

}

if (i <= 0)

{

printf("Fuction \"Select\" can't find this value");

return NULL;

}

}

else if ((strcmp(field, "x\_size")) == 0)

{

while (fread(db\_point, sizeof(SCAN\_INFO), 1, DB))

if (db\_point->x\_size == atoi(value))

rec->rec\_nmb++;

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

{

free(db\_point);

free(rec);

fclose(DB);

return NULL;

}

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

while (fread(db\_point, sizeof(SCAN\_INFO), 1, DB))

if (db\_point->x\_size == atoi(value))

{

(rec->recs[i]) = \*db\_point;

i++;

}

if (i <= 0)

{

printf("Fuction \"Select\" can't find this value");

return NULL;

}

}

else if ((strcmp(field, "y\_size")) == 0)

{

while (fread(db\_point, sizeof(SCAN\_INFO), 1, DB))

if (db\_point->y\_size == atoi(value))

rec->rec\_nmb++;

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

{

free(db\_point);

free(rec);

fclose(DB);

return NULL;

}

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

while (fread(db\_point, sizeof(SCAN\_INFO), 1, DB))

if (db\_point->y\_size == atoi(value))

{

(rec->recs[i]) = \*db\_point;

i++;

}

if (i <= 0)

{

printf("Fuction \"Select\" can't find this value");

return NULL;

}

}

fclose(DB);

return rec;

}

void reindex(const char \*db)

{

make\_index(db, "manufacturer.idx");

make\_index(db, "year.idx");

make\_index(db, "model.idx");

make\_index(db, "price.idx");

make\_index(db, "x\_size.idx");

make\_index(db, "y\_size.idx");

}

void print\_rec\_set(RECORD\_SET \*rs)

{

int i;

if (rs == NULL) return;

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

printf("%s;%s;%d;%.2f;%d;%d\n", (rs->recs[i].manufacturer), (rs->recs[i].model), rs->recs[i].year, rs->recs[i].price, rs->recs[i].x\_size, rs->recs[i].y\_size);

}

void add\_scanner(const char \*db, const char\* scanner\_str)

{

FILE \*DB;

SCAN\_INFO \*db\_point;

int numbersOfStructs;

db\_point = (SCAN\_INFO\*)malloc(sizeof(SCAN\_INFO));

if (db\_point == NULL) return;

if ((DB = fopen(db, "rb+")) == NULL)

exit(1);

fread(&numbersOfStructs, sizeof(int), 1, DB);

numbersOfStructs +=1;

fseek(DB, 0, SEEK\_SET);

fwrite(&numbersOfStructs, sizeof(int), 1, DB);

sscanf(scanner\_str, "%[A-z0-9];%[A-z0-9];%d;%f;%d;%d", &(db\_point->manufacturer), &(db\_point->model), &(db\_point->year), &(db\_point->price), &(db\_point->x\_size), &(db\_point->y\_size));

fseek(DB, 0, SEEK\_END);

fwrite(db\_point, sizeof(SCAN\_INFO), 1, DB);

fclose(DB);

}

void del\_scanner(const char \*db, int id)

{

FILE \*DB;

SCAN\_INFO \*db\_point;

int numbersOfStructs, offsetStructs = -2;

DB = fopen(db, "rb+");

if (DB == NULL) return;

fread(&numbersOfStructs, sizeof(int), 1, DB);

numbersOfStructs -= 1;

fseek(DB, 0, SEEK\_SET);

fwrite(&numbersOfStructs, sizeof(int), 1, DB);

db\_point = (SCAN\_INFO\*)malloc(sizeof(SCAN\_INFO));

if (db\_point == NULL) return;

fseek(DB, (id + 1) \* sizeof(SCAN\_INFO) + sizeof(int), SEEK\_SET);

while (fread(db\_point, sizeof(SCAN\_INFO), 1, DB))

{

fseek(DB, offsetStructs \* sizeof(SCAN\_INFO), SEEK\_CUR);

fwrite(db\_point, sizeof(SCAN\_INFO), 1, DB);

fseek(DB, sizeof(SCAN\_INFO), SEEK\_CUR);

}

\_chsize(\_fileno(DB), numbersOfStructs \* sizeof(SCAN\_INFO) + sizeof(int));

free(db\_point);

fclose(DB);

}

TestScanner.c

#include "Scanner.h"

int main()

{

printf("\t\t\tDATA BASE LAB - 2\n\tFirst List of Scanners CSV FILE:\n");

create\_db("Scanners.csv", "db.bin");

printf("\n\t First List of Scanners DB file:\n");

print\_db("db.bin");

reindex("db.bin");

printf("\nAdd scanner: \"ASUS;AS1055;2016;199.99;1210;1997\"\n");

add\_scanner("db.bin", "ASUS;AS1055;2016;199.99;1210;1997");

print\_db("db.bin");

reindex("db.bin");

printf("\nDell scanner with 5 id: \n");

del\_scanner("db.bin", 5);

print\_db("db.bin");

reindex("db.bin");

printf("\nFuction Select: \n");

RECORD\_SET\* rs = select("db.bin", "year", "2016");

print\_rec\_set(rs);

printf("\n\tIDX file\_name: \n");

readIDX("manufacturer.idx");

printf("\tFunction recs\_by\_index: \n");

rs = get\_recs\_by\_index("db.bin", "manufacturer.idx");

print\_rec\_set(rs);

\_getch();

return 0;

}

**Тест**

