**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ  
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ   
імені ІГОРЯ СІКОРСЬКОГО»**

**Інститут прикладного системного аналізу  
Кафедра системного проектування**

**Звіт**

**про виконання лабораторної роботи №14   
з дисципліни “Алгоритми та програмування”**

Виконав:  
студент I курсу, групи ДА-12  
Кракович Павло Дмитрович

Прийняв:

к.т.н., доцент Безносик О. Ю.

Київ – 2022

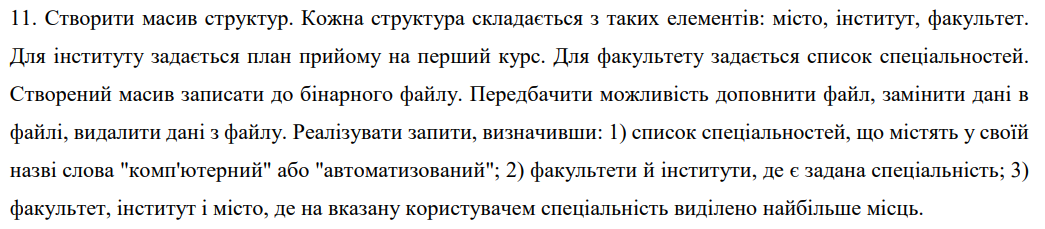
1.  Скласти алгоритм і програму рішення запропонованої задачі (рис. 1)

Рис. 1

1. Аналіз умови задачі.

Для виконання завдання потрібно створити структури для зберігання даних та методи роботи з ними. Також, слід додати меню для зручного користування програмою. За можливості, всі методи та структури додатково слід винести у окремий заголовочний файл.

1. Код

* main.c

#include <stdio.h>  
#include <string.h>  
#include "utility.h"  
  
int main() {  
 char\* filepath = strcat(path, filename);  
 clearFile(readbuffer, filepath);  
  
 struct Object tempData[1];  
 struct Object data[32];  
  
 int user\_input,index,numberOfElements=0;  
 char dataToFind[16],answer;  
 printf(  
 "\tMenu\n"  
 "[ 1 ] Add data\n"  
 "[ 2 ] Edit data\n"  
 "[ 3 ] Remove data\n"  
 "[ 4 ] Display data\n"  
 "[ 5 ] Find data (keyword)\n"  
 "[ 6 ] Find data (by code)\n"  
 "[ 7 ] Find biggest size\n"  
 "[ 8 ] Fill file with data\n"  
 "[ 9 ] Clear all data\n");  
 do {  
 user\_input = getchar();  
 switch (user\_input) {  
 case '1':  
 do {  
 printf("Town:");  
 scanf("%s", data[0].town.name);  
 printf("University:");  
 scanf("%s", data[0].university.name);  
 printf("Speciality code:");  
 scanf("%d", &data[0].university.speciality);  
 printf("Faculty:");  
 scanf("%s", data[0].faculty.name);  
 printf("Speciality:");  
 getchar(); // Clear buffer  
 gets(data[0].faculty.speciality);  
 printf("Size:");  
 scanf("%d", &data[0].faculty.size);  
  
 printf("continue? (y/n)\n");  
 do {  
 answer = getchar();  
 } while (answer != 'y' && answer != 'n' && answer != 'Y' && answer != 'N');  
 writeData(writebuffer, filepath, &data[0], 1);  
 numberOfElements++;  
 } while (answer != 'n' && answer != 'N');  
 break;  
 case '2':  
 if (numberOfElements == 0) {  
 printf("File is empty:\n");  
 break;  
 }  
 printf("Type index of the element:\n");  
 scanf("%d", &index);  
 if (index > numberOfElements || index < 0) {  
 printf("Out of range.\n");  
 break;  
 }  
 printf("Town:");  
 scanf("%s", tempData[0].town.name);  
 printf("University:");  
 scanf("%s", tempData[0].university.name);  
 printf("Speciality code:");  
 scanf("%d", &tempData[0].university.speciality);  
 printf("Faculty:");  
 scanf("%s", tempData[0].faculty.name);  
 printf("Speciality:");  
 getchar(); // Clear buffer  
 gets(tempData[0].faculty.speciality);  
 printf("Size:");  
 scanf("%d", &tempData[0].faculty.size);  
 editData(writebuffer, filepath, tempData, index);  
 break;  
 case '3':  
 if (numberOfElements == 0) {  
 printf("File is empty:\n");  
 break;  
 }  
 printf("Enter index to delete:\n");  
 scanf("%d", &index);  
 deleteData(readbuffer, filepath, index);  
 numberOfElements--;  
 break;  
 case '4':  
 if (numberOfElements == 0) {  
 printf("File is empty.\n");  
 break;  
 }  
 printData(readbuffer, filepath);  
 break;  
 case '5':  
 if (numberOfElements == 0) {  
 printf("File is empty.\n");  
 break;  
 }  
 printf("Enter keyword:");  
 scanf("%s",dataToFind);  
 findData(readbuffer, filepath, dataToFind);  
 break;  
 case '6':  
 if (numberOfElements == 0) {  
 printf("File is empty.\n");  
 break;  
 }  
 printf("Enter speciality code:");  
 scanf("%d",&index);  
 printDataByNumber(readbuffer, filepath, index);  
 break;  
 case '7':  
 if (numberOfElements == 0) {  
 printf("File is empty.\n");  
 break;  
 }  
 findBiggestSize(readbuffer, filepath);  
 break;  
 case '8':  
 clearAndFillFile(readbuffer, filepath);  
 printf("File filled.\n");  
 numberOfElements = 5;  
 break;  
 case '9':  
 clearFile(readbuffer, filepath);  
 numberOfElements = 0;  
 printf("File cleared.\n");  
 break;  
 default:;  
 }  
 }  
 while (user\_input != 0);  
 return 0;  
}

* utility.h

#ifndef LABA14\_UTILITY\_H  
#define LABA14\_UTILITY\_H  
  
#include <stdio.h>  
#include <string.h>  
  
FILE\* writebuffer;  
FILE\* readbuffer;  
  
char path[32] = "C:/labsData/";  
char filename[16] = "data.bin";  
  
struct Town {  
 char name[16];  
};  
  
struct University {  
 char name[16];  
 int speciality;  
};  
  
struct Faculty {  
 char name[32];  
 char speciality[64];  
 int size;  
};  
  
struct Object {  
 struct Town town;  
 struct University university;  
 struct Faculty faculty;  
};  
  
void writeData(FILE\* writebuffer, const char\* filepath, struct Object\* data, size\_t len)  
{  
 writebuffer = fopen(filepath, "ab");  
 for (size\_t i = 0; i < len; i++)  
 {  
 fwrite(&data[i], sizeof(struct Object), 1, writebuffer);  
 }  
 fclose(writebuffer);  
}  
  
void printData(FILE\* readbuffer, const char\* filepath)  
{  
 struct Object output;  
 readbuffer = fopen(filepath, "rb");  
  
 while (fread(&output, sizeof(output), 1, readbuffer) == 1)  
 {  
 printf("Town: %s, University: %s, Code: %d, Faculty: %s, Speciality: %s, Size: %d;\n",  
 output.town.name,  
 output.university.name,  
 output.university.speciality,  
 output.faculty.name,  
 output.faculty.speciality,  
 output.faculty.size  
 );  
 }  
 fclose(readbuffer);  
}  
  
void editData(FILE\* restrict writebuffer, const char\* filepath, struct Object\* data, int i)  
{  
 writebuffer = fopen(filepath, "rb+");  
 fseek(writebuffer, sizeof(struct Object) \* i, SEEK\_SET);  
 fwrite(data, sizeof(struct Object), 1, writebuffer);  
 fclose(writebuffer);  
}  
  
void deleteData(FILE\* restrict readbuffer, const char\* filepath, int i)  
{  
 FILE \*readbuffer\_tmp;  
 int found=0;  
 struct Object output;  
  
 readbuffer=fopen(filepath, "rb");  
 if (!readbuffer) {  
 printf("Unable to open file %s", filepath);  
 return;  
 }  
 readbuffer\_tmp=fopen("tmp.bin", "rb");  
 if (!readbuffer\_tmp) {  
 printf("Unable to open file temp file.");  
 return;  
 }  
  
 for (int j = 0; fread(&output, sizeof(struct Object), 1, readbuffer) == 1; j++)  
 {  
 if (i == j)  
 {  
 printf("Successfully deleted record at index %d\n", i);  
 found = 1;  
 }  
 else fwrite(&output, sizeof(struct Object), 1, readbuffer\_tmp);  
 }  
 if (!found) printf("Couldn't find a record at index %d\n", i);  
  
 fclose(readbuffer);  
 fclose(readbuffer\_tmp);  
  
 remove(filepath);  
 rename("tmp.bin", filepath);  
}  
  
void findData(FILE\* restrict readbuffer, const char\* filepath, const char\* data) {  
 struct Object output;  
 readbuffer = fopen(filepath, "rb");  
 while (fread(&output, sizeof(output), 1, readbuffer) == 1)  
 {  
 if (strstr(output.faculty.speciality, data) != NULL) {  
 printf("Town: %s, University: %s, Faculty: %s, Speciality: %s;\n",  
 output.town.name, output.university.name, output.faculty.name, output.faculty.speciality);  
 }  
 }  
 fclose(readbuffer);  
}  
  
void findBiggestSize(FILE\* readbuffer, const char\* filepath) {  
 struct Object output;  
 readbuffer = fopen(filepath, "rb");  
 int maxNumber = 0;  
 char speciality[32];  
 while (fread(&output, sizeof(output), 1, readbuffer) == 1)  
 {  
 if (maxNumber < output.faculty.size)  
 {  
 maxNumber = output.faculty.size;  
 strcpy(speciality,output.faculty.speciality);  
 }  
 }  
 printf("Speciality with largest number of places: %s\n", speciality);  
 fclose(readbuffer);  
}  
  
void printDataByNumber(FILE\* readbuffer, const char\* filepath, int index) {  
 struct Object output;  
 readbuffer = fopen(filepath, "rb");  
 while (fread(&output, sizeof(output), 1, readbuffer) == 1)  
 {  
 if (index == output.university.speciality)  
 {  
 printf("Town: %s, University: %s, Faculty: %s;\n", output.town.name, output.university.name, output.faculty.name);  
 }  
 }  
 fclose(readbuffer);  
}  
  
void clearFile(FILE\* readbuffer,const char\* filepath) {  
 FILE \*tmp;  
  
 readbuffer=fopen(filepath, "rb");  
 if (!readbuffer) {  
 printf("Unable to open file %s", filepath);  
 return;  
 }  
 tmp=fopen("tmp.bin", "wb");  
 if (!tmp) {  
 printf("Unable to open file temp file.");  
 return;  
 }  
  
 fclose(readbuffer);  
 fclose(tmp);  
  
 remove(filepath);  
 rename("tmp.bin", filepath);  
}  
  
void clearAndFillFile(FILE\* readbuffer,const char\* filepath) {  
 clearFile(readbuffer, filepath);  
 struct Object data[32] = {  
 {  
 {"Kiev"},  
 {"KPI",122},  
 {"IASA", "System Design", 85}  
 },  
 {  
 {"Kiev"},  
 {"KPI",124},  
 {"IASA", "Math. Methods of System Analysis", 70}  
 },  
 {  
 {"Kiev"},  
 {"KPI",122},  
 {"FBMI", "Biomedical Cybernetics", 60}  
 },  
 {  
 {"Kiev"},  
 {"KPI",122},  
 {"EIT", "Computer Science", 95}  
 },  
 {  
 {"Kiev"},  
 {"KPI",121},  
 {"EIT", "Software Engineering", 105}  
 }  
 };  
 writeData(writebuffer, filepath, &data[0], 5);  
}  
  
  
#endif //LABA14\_UTILITY\_H

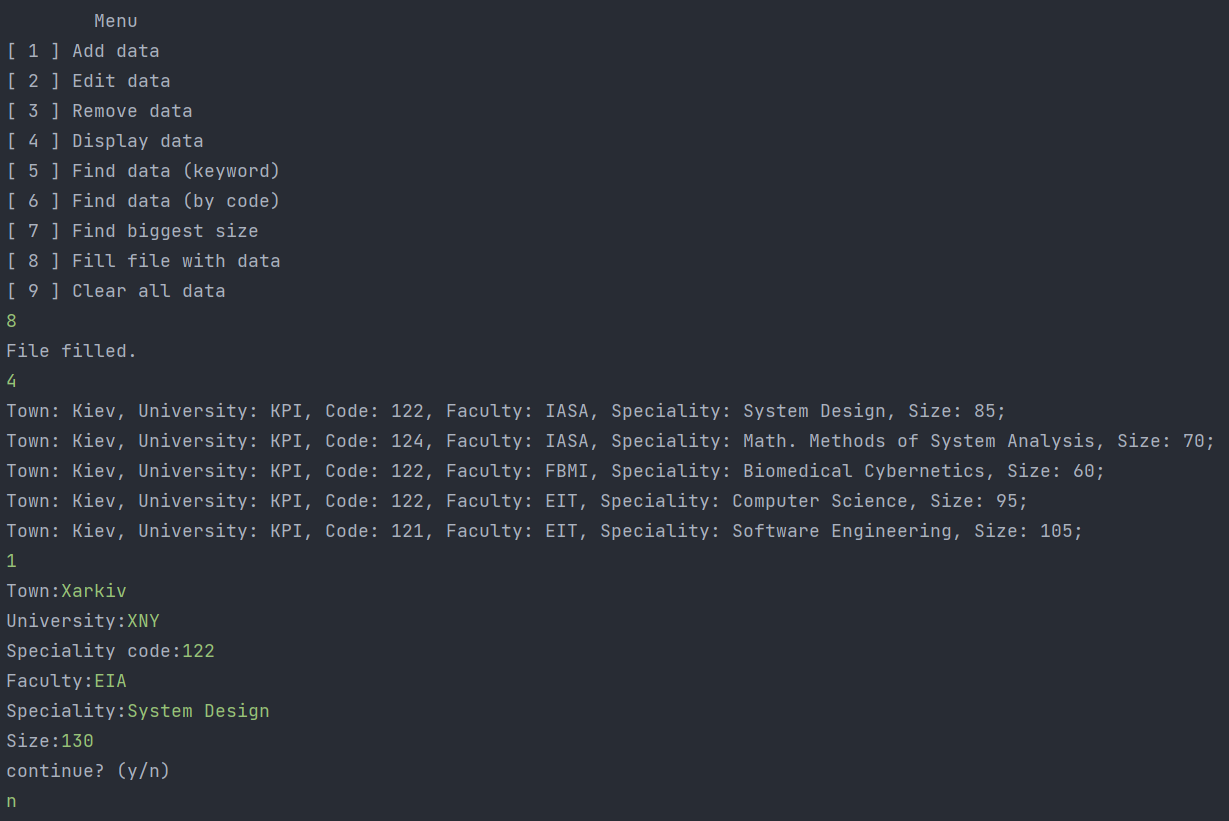
1. Результат роботи програми: (рис. 2-5)

Рис. 2

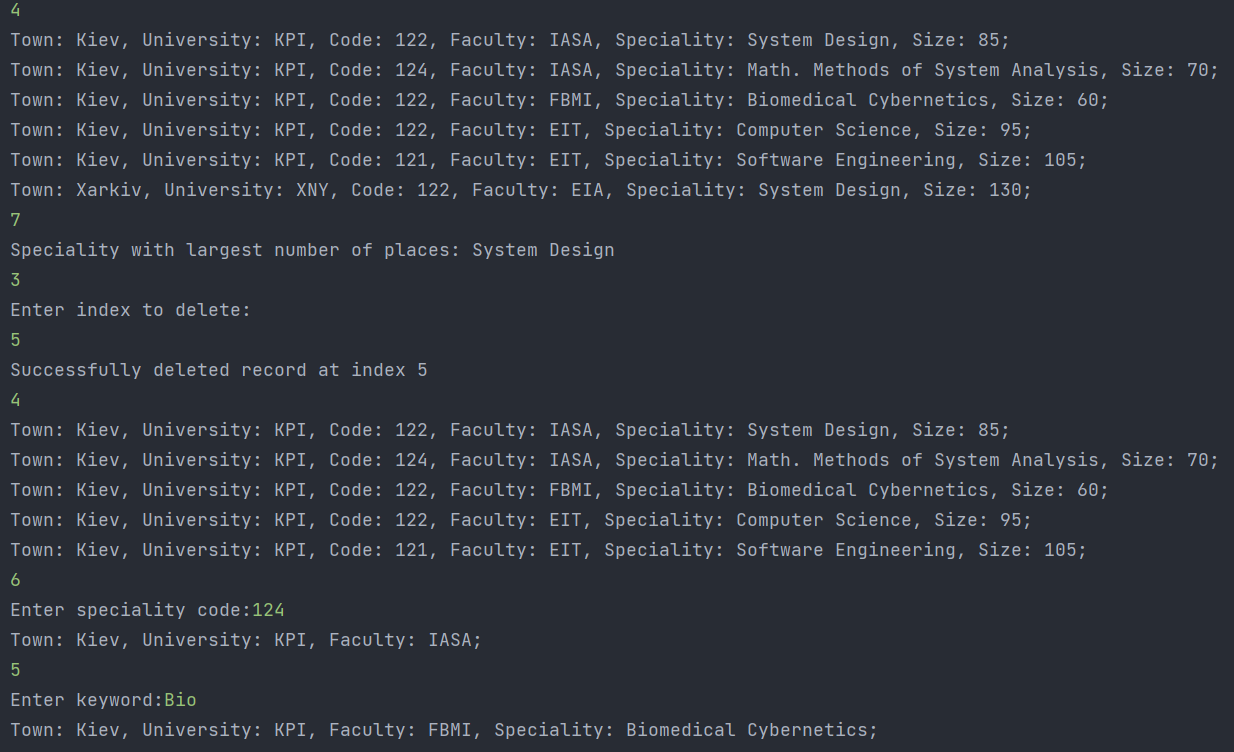


Рис. 3

Висновок

В ході виконання лабораторної роботи №14 я створив структуру даних для та методи для роботи з нею. Також, я реалізував окрім стандартних методів роботи зі структурою(додавання, видалення та редагування) ще методи для пошуку спеціальності по назві, пошуку спеціальностей по їх коду та пошук спеціальності з найбільшим замовленням місць на 1 курс.

Виконуючи лабораторну роботу я вивчив особливості використання структур, об’єднань та файлів та навчитися застосовувати структури, об’єднання, файли в програмуванні.