**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

Лабораторная работа №2

По дисциплине «Программирование»

 Массивы структур

Группа: АВТ-041

Студент: Дударь М.А.

Преподаватель: Балакин В.В.

НОВОСИБИРСК 2021

1. **Постановка задачи:**

Необходимо определить структурированный тип и набор функций для работы с таблицей записей, реализованной в массиве структур. В перечень функций входят:

* ввод записи таблицы с клавиатуры;
* загрузка и сохранение таблицы в текстовом файле;
* просмотр таблицы;
* сортировка таблицы в порядке возрастания заданного поля;
* поиск в таблице элемента с заданным значением поля или с наиболее близким к нему по значению;
* удаление записи;
* изменение (редактирование) записи;
* вычисление с проверкой и использованием всех записей по заданному условию и формуле (например, общая сумма на всех счетах).

Перечень полей структурированной переменной: фамилия, дата поступления, дата отчисления.

1. **Алгоритм программы:**
2. Считать данные из консоли или файла.
3. Записать данные в файл.
4. Просмотреть данные.
5. Отсортировать данные.
6. Найти определенные данные.
7. Удалить определенные данные.
8. Изменить определенные данные.
9. Вычислить средний срок обучения.
10. **Методы решения:**

Для решения задачи использовался массив структур. Он удобен тем, что в один объект можно записать совокупность данных, к которым в дальнейшем можно будет обратиться.

1. **Принцип работы:**

Структура, хранящая имя объекта, массив даты поступления и массив даты отчисления.

struct table {  
 char firstName[20];  
 int res[3];  
 int exp[3];  
};

Функция проверки введенных данных.

void input\_validation(struct table new, int \*flag) {  
 if (new.res[0] > 31 || new.exp[0] > 31 || new.res[0] < 1 || new.exp[0] < 1  
 || new.res[1] > 12 || new.exp[1] > 12 || new.res[1] < 1 || new.exp[1] < 1  
 || new.res[2] > 2021 || new.exp[2] > 2021 || new.res[2] < 1900 || new.exp[2] < 1900  
 || new.exp[2] > new.res[2] || (new.exp[1] == 2) && (new.exp[0] == 30) ||  
 (new.res[1] == 2) && (new.res[0] == 30)) {  
 printf("Unavailable input\n");  
 \*flag = 1;  
 }  
}

Функция чтения объектов из консоли.

struct table Input(int \*flag) {  
 struct table new;  
 printf("Enter Family\n");  
 scanf("%s", new.firstName);  
 printf("Enter receipt date (dd mm yy)\n");  
 scanf("%d", &new.res[0]);  
 scanf("%d", &new.res[1]);  
 scanf("%d", &new.res[2]);  
 printf("Enter date of expulsion (dd mm yy)\n");  
 scanf("%d", &new.exp[0]);  
 scanf("%d", &new.exp[1]);  
 scanf("%d", &new.exp[2]);  
 input\_validation(new, flag);  
 return (new);  
}

Функция вывода объекта в консоль.

void Output(struct table new) {  
 printf("First name is %s\n", new.firstName);  
 printf("Receipt date is %d %d %d\n", new.res[0], new.res[1], new.res[2]);  
 printf("Date of expulsion is %d %d %d\n\n", new.exp[0], new.exp[1], new.exp[2]);  
}

Функция чтения объектов из файла.

struct table \*File\_read(int \*flag, int \*len) {  
 struct table \*new = (struct table \*) malloc(2 \* sizeof(struct table));  
 FILE \*fp;  
 if ((fp = fopen("read.txt", "r")) == NULL) {  
 printf("Can't open file 'read.txt'\n");  
 \*flag = 1;  
 }  
 while (fscanf(fp, "%s", new[\*len].firstName) == 1) {  
 fscanf(fp, "%\*s%\*s%d %d %d", &new[\*len].res[0], &new[\*len].res[1], &new[\*len].res[2]);  
 fscanf(fp, "%\*s%\*s%\*s%d %d %d\n", &new[\*len].exp[0], &new[\*len].exp[1], &new[\*len].exp[2]);  
 new = (struct table \*) realloc(new, (\*len + 2) \* sizeof(struct table[2]));  
 (\*len)++;  
 }  
 fclose(fp);  
 for (int i = 0; i < \*len; ++i) input\_validation(new[i], flag);  
 return new;  
}

Функция записи объектов в файл.

void File\_write(struct table new, int \*flag, int count, char \*name) {  
 FILE \*fp;  
 if ((fp = fopen(name, "r+w")) == NULL) {  
 printf("Can't open file 'write.txt'\n");  
 \*flag = 1;  
 }  
 struct table buf;  
 for (int i = 0; i < count; i++) {  
 fscanf(fp, "%s", buf.firstName);  
 fscanf(fp, "%\*s%\*s%d %d %d", &buf.res[0], &buf.res[1], &buf.res[2]);  
 fscanf(fp, "%\*s%\*s%\*s%d %d %d\n", &buf.exp[0], &buf.exp[1], &buf.exp[2]);  
 }  
 fprintf(fp, "%s\n", new.firstName);  
 fprintf(fp, "Receipt date: %d %d %d\n", new.res[0], new.res[1], new.res[2]);  
 fprintf(fp, "Date of expulsion: %d %d %d\n", new.exp[0], new.exp[1], new.exp[2]);  
 fclose(fp);  
}

Функция поиска в массиве структур числового значения.

int Find\_int(struct table \*new, int count, int value) {  
 int delta[2] = {INT\_MAX, 0};  
 for (int i = 0; i < count; ++i) {  
 for (int j = 0; j < 3; ++j) {  
 if (new[i].res[j] == value) return new[i].res[j];  
 if (new[i].exp[j] == value) return new[i].exp[j];  
 if (abs(value - new[i].res[j]) < delta[0]) {  
 delta[0] = abs(value - new[i].res[j]);  
 delta[1] = i;  
 }  
 if (abs(value - new[i].exp[j]) < delta[0]) {  
 delta[0] = abs(value - new[i].exp[j]);  
 delta[1] = i;  
 }  
 }  
 }  
 for (int i = 0; i < count; ++i) {  
 for (int j = 0; j < 3; ++j) {  
 if (abs(value - new[delta[1]].exp[j]) == delta[0]) {  
 printf("%s\nExpulsion date = %d\n", new[delta[1]].firstName, new[delta[1]].exp[j]);  
 return 0;  
 }  
 if (abs(value - new[delta[1]].res[j]) == delta[0]) {  
 printf("%s\nReceipt date = %d\n", new[delta[1]].firstName, new[delta[1]].res[j]);  
 return 0;  
 }  
 }  
 }  
 return 0;  
}

Функция поиска в массиве структур строки.

int Find\_char(struct table \*new, int count, char \*value) {  
 int match, rez = 0;  
 for (int i = 0; i < count; ++i) {  
 match = 0;  
 for (int j = 0; j < strlen(value); ++j) {  
 if (new[i].firstName[0] == value[j]) match++;  
 }  
 if (match > rez) rez = match;  
 }  
 for (int i = 0; i < count; ++i) {  
 match = 0;  
 for (int j = 0; j < strlen(value); ++j) {  
 if (new[i].firstName[0] == value[j]) match++;  
 }  
 if (match == rez) {  
 printf("%s\n", new[i].firstName);  
 return 0;  
 }  
 }  
 return 0;  
}

}

Функция сортировки массива структур по алфавиту, возрастанию даты поступления или отчисления.

void Sort(struct table \*new, int count, int select) {  
 int i, j;  
 struct table temp;  
 for (i = 0; i < count - 1; i++)  
 for (j = 0; j < count - i - 1; j++) {  
 for (int k = 0; k < 3; ++k) {  
 switch (select) {  
 case 1:  
 if ((int) new[j].firstName[0] > (int) new[j + 1].firstName[0]) {  
 temp = new[j];  
 new[j] = new[j + 1];  
 new[j + 1] = temp;  
  
 }  
 break;  
 case 2:  
 if (new[j].res[k] > new[j + 1].res[k]) {  
 temp = new[j];  
 new[j] = new[j + 1];  
 new[j + 1] = temp;  
  
 }  
 break;  
 case 3:  
 if (new[j].exp[k] > new[j + 1].exp[k]) {  
 temp = new[j];  
 new[j] = new[j + 1];  
 new[j + 1] = temp;  
  
 }  
 break;  
 default:  
 printf("Error\n");  
 break;  
 }  
 }  
 }  
 for (int k = 0; k < count; ++k)  
 File\_write(new[k], 0, k, "sorted.txt");  
}

Функция удаления выбранного поля объекта.

void Delete\_value(struct table \*new, const int \*count) {  
 if (count[1] == 1) strcpy(new[count[0]].firstName, "");  
 if (count[1] == 2) new[count[0]].res[count[2]] = 0;  
 if (count[1] == 3) new[count[0]].exp[count[2]] = 0;  
  
}

Функция удаления выбранного объекта из массива структур.

struct table \*Delete\_object(struct table \*new, const int \*count, int \*len) {  
 for (int i = \*count; i < \*len; ++i) {  
 printf("%s<->", new[i].firstName);  
 new[i] = new[i + 1];  
 printf("%s\n", new[i].firstName);  
 }  
 (\*len)--;  
 new = (struct table \*) realloc(new, \*len \* sizeof(struct table));  
 return new;  
}

Функция изменения выбранного поля объекта.

int Edit\_value(struct table \*new, const int \*count, int iValue, char \*cValue) {  
 if (count[1] == 1) strcpy(new[count[0]].firstName, cValue);  
 if (count[1] == 2) new[count[0]].res[count[2]] = iValue;  
 if (count[1] == 3) new[count[0]].exp[count[2]] = iValue;  
 return 0;  
}

Функция опроса, к какому полю обратиться.

int Interface(struct table \*a, int len, int \*select\_id, int bool, int \*iValue, char \*cValue) {  
 for (int i = 0; i < len; ++i) {  
 printf("#%d - %s\n", i + 1, a[i].firstName);  
 }  
 printf("Chose user number: ");  
 scanf("%d", &select\_id[0]);  
 select\_id[0]--;  
 if ((select\_id[0] < 1) || (select\_id[0] > len + 1)) {  
 printf("Unavailable input\n");  
 return 1;  
 }  
 printf("#1 %s\n"  
 "#2 %d %d %d\n"  
 "#3 %d %d %d\n", a[select\_id[0]].firstName, a[select\_id[0]].res[0], a[select\_id[0]].res[1],  
 a[select\_id[0]].res[2], a[select\_id[0]].exp[0], a[select\_id[0]].exp[1], a[select\_id[0]].exp[2]);  
  
 printf("Select the field: ");  
 scanf("%d", &select\_id[1]);  
 if ((select\_id[1] < 1) || (select\_id[1] > 3)) {  
 printf("Unavailable input\n");  
 return 1;  
 }  
 if (select\_id[1] == 1) {  
 Delete\_value(a, select\_id);  
 if (bool == 1) {  
 printf("Enter a value: ");  
 scanf("%s", cValue);  
 }  
 } else {  
 printf("Select cell\n#1 dd\n#2 mm\n#3 yy\n");  
 scanf("%d", &select\_id[2]);  
 select\_id[2]--;  
 if ((select\_id[2] < 1) || (select\_id[2] > 3)) {  
 printf("Unavailable input\n");  
 return 1;  
 }  
 if (bool == 1) {  
 printf("Enter a value: ");  
 scanf("%d", iValue);  
 }  
 }  
 return 0;  
}

Функция вычисления среднего значения длительности обучения.

void Average(struct table \*new, int len) {  
 double delta[3];  
 for (int j = 0; j < 3; ++j) delta[j] = 0;  
 for (int i = 0; i < len; ++i)  
 for (int j = 0; j < 3; ++j)  
 delta[j] += abs(new[i].res[j] - new[i].exp[j]);  
 for (int j = 0; j < 3; ++j) {  
 delta[j] /= len;  
 printf("%.2f ", round(delta[j] \* 1000) / 1000);  
 }  
 printf("\n");  
}

Функция проверки интервала введенной переменной.

int check(int select, int a, int b) {  
 if (select < a || select > b) {  
 printf("Unavailable input\n");  
 return 1;  
 } else return 0;  
}

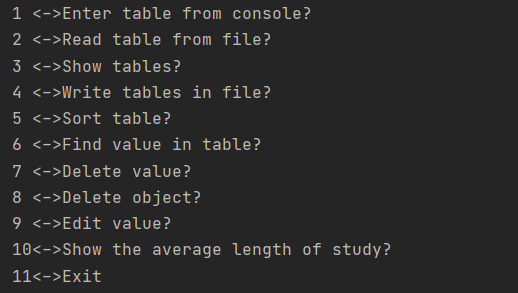
Функция меню выбора необходимых действий с массивом указателей.

int Ask() {  
 int flag = 0, select, fInt, len = 0, rez, iValue, buf= 0;  
 int select\_id[3];  
 char \*cValue = (char \*) malloc(20 \* sizeof(char));  
 char fChar[20];  
 struct table \*a;  
 while (1) {  
 printf("1 <->Enter table from console?\n"  
 "2 <->Read table from file?\n"  
 "3 <->Show tables?\n"  
 "4 <->Write tables in file?\n"  
 "5 <->Sort table?\n"  
 "6 <->Find value in table?\n"  
 "7 <->Delete value?\n"  
 "8 <->Delete object?\n"  
 "9 <->Edit value?\n"  
 "10<->Show the average length of study?\n"  
 "11<->Exit\n");  
 scanf("%d", &select);  
 if (check(select, 1, 12)) return 0;  
 if ((len == 0) && ((select == 3) || (select > 4) && (select < 11))) {  
 printf("\tVoid table. Add objects!\n\n");  
 continue;  
 }  
 switch (select) {  
 case 1:  
 printf("How many table do you want to enter?\n");  
 scanf("%d", &buf);  
 len+=buf;  
 a = (struct table \*) realloc(a, sizeof(struct table) \* len);  
 for (int i = len - buf; i < len; ++i) {  
 a[i] = Input(&flag);  
 if (flag == 1) return 0;  
 }  
 break;  
 case 2:  
 a = File\_read(&flag, &len);  
 printf("Successful!\n");  
 break;  
 case 3:  
 for (int i = 0; i < len; ++i) Output(a[i]);  
 break;  
 case 4:  
 for (int i = 0; i < len; ++i) File\_write(a[i], &flag, i, "write.txt");  
 break;  
 case 5:  
 printf("Select sort:\n"  
 "1-Sort by last name\n"  
 "2-Sort by date of receipt\n"  
 "3-Sort by expiration date\n");  
 scanf("%d", &select);  
 if (check(select, 1, 3)) return 0;  
 Sort(a, len, select);  
 for (int i = 0; i < len; ++i) {  
 printf("%s\n", a[i].firstName);  
 switch (select) {  
 case 1:  
 continue;  
 case 2:  
 printf("%d %d %d\n", a[i].res[0], a[i].res[1], a[i].res[2]);  
 break;  
 case 3:  
 printf("%d %d %d\n", a[i].exp[0], a[i].exp[1], a[i].exp[2]);  
 break;  
 default:  
 return 0;  
 }  
 }  
 break;  
 case 6:  
 printf("What do you want to find int or char?\n"  
 "1-Int\n"  
 "2-Char\n");  
 scanf("%d", &select);  
 if (check(select, 1, 2)) return 0;  
 printf("Enter value\n");  
 switch (select) {  
 case 1:  
 scanf("%d", &fInt);  
 Find\_int(a, len, fInt);  
 break;  
 case 2:  
 scanf("%s", fChar);  
 Find\_char(a, len, fChar);  
 break;  
 default:  
 printf("Unavailable input\n");  
 return 0;  
 }  
 break;  
 case 7:  
 rez = Interface(a, len, select\_id, 0, 0, 0);  
 if (rez == 1) return 0;  
 if (rez == 2) Delete\_value(a, select\_id);  
 printf("#1 %s\n"  
 "#2 %d %d %d\n"  
 "#3 %d %d %d\n", a[select\_id[0]].firstName, a[select\_id[0]].res[0], a[select\_id[0]].res[1],  
 a[select\_id[0]].res[2], a[select\_id[0]].exp[0], a[select\_id[0]].exp[1],  
 a[select\_id[0]].exp[2]);  
 break;  
 case 8:  
 for (int i = 0; i < len; ++i) {  
 printf("####%d####\n", i);  
 printf("%s\n", a[i].firstName);  
 }  
 printf("Enter the object number to be deleted\n");  
 scanf("%d", &select\_id[2]);  
 if (check(select\_id[2], 1, len)) return 0;  
 a = Delete\_object(a, &select\_id[2], &len);  
 for (int i = 0; i < len; ++i) {  
 printf("####%d####\n", i);  
 printf("%s\n", a[i].firstName);  
 }  
 break;  
 case 9:  
 rez = Interface(a, len, select\_id, 1, &iValue, cValue);  
 if (rez == 1) return 0;  
 Edit\_value(a, select\_id, iValue, cValue);  
 printf("\n%s\n"  
 "%d %d %d\n"  
 "%d %d %d\n", a[select\_id[0]].firstName, a[select\_id[0]].res[0], a[select\_id[0]].res[1],  
 a[select\_id[0]].res[2], a[select\_id[0]].exp[0], a[select\_id[0]].exp[1],  
 a[select\_id[0]].exp[2]);  
 break;  
 case 10:  
 printf(" dd mm yy\n");  
 Average(a, len);  
 break;  
 case 11:  
 return 0;  
 default:  
 printf("Unavailable input\n");  
 }  
 }  
}

1. **Вывод:**

В ходе выполнения задания была произведена работа со структурами, файлами на языке C. Создан массив структур, реализовано взаимодействие с ними, динамическое изменение их размерности. Применена работа с файлами.

Приложение 1. Вывод программы.



Приложение 2. Текст программы.

#include <stdio.h>  
#include <malloc.h>  
#include <limits.h>  
#include <string.h>  
#include <math.h>  
  
struct table {  
 char firstName[20];  
 int res[3];  
 int exp[3];  
};  
  
void input\_validation(struct table new, int \*flag) {  
 if (new.res[0] > 31 || new.exp[0] > 31 || new.res[0] < 1 || new.exp[0] < 1  
 || new.res[1] > 12 || new.exp[1] > 12 || new.res[1] < 1 || new.exp[1] < 1  
 || new.res[2] > 2021 || new.exp[2] > 2021 || new.res[2] < 1900 || new.exp[2] < 1900  
 || new.exp[2] > new.res[2] || (new.exp[1] == 2) && (new.exp[0] == 30) ||  
 (new.res[1] == 2) && (new.res[0] == 30)) {  
 printf("Unavailable input\n");  
 \*flag = 1;  
 }  
}  
  
struct table Input(int \*flag) {  
 struct table new;  
 printf("Enter Family\n");  
 scanf("%s", new.firstName);  
 printf("Enter receipt date (dd mm yy)\n");  
 scanf("%d", &new.res[0]);  
 scanf("%d", &new.res[1]);  
 scanf("%d", &new.res[2]);  
 printf("Enter date of expulsion (dd mm yy)\n");  
 scanf("%d", &new.exp[0]);  
 scanf("%d", &new.exp[1]);  
 scanf("%d", &new.exp[2]);  
 input\_validation(new, flag);  
 return (new);  
}  
  
void Output(struct table new) {  
 printf("First name is %s\n", new.firstName);  
 printf("Receipt date is %d %d %d\n", new.res[0], new.res[1], new.res[2]);  
 printf("Date of expulsion is %d %d %d\n\n", new.exp[0], new.exp[1], new.exp[2]);  
}  
  
void File\_write(struct table new, int \*flag, int count, char \*name) {  
 FILE \*fp;  
 if ((fp = fopen(name, "r+w")) == NULL) {  
 printf("Can't open file 'write.txt'\n");  
 \*flag = 1;  
 }  
 struct table buf;  
 for (int i = 0; i < count; i++) {  
 fscanf(fp, "%s", buf.firstName);  
 fscanf(fp, "%\*s%\*s%d %d %d", &buf.res[0], &buf.res[1], &buf.res[2]);  
 fscanf(fp, "%\*s%\*s%\*s%d %d %d\n", &buf.exp[0], &buf.exp[1], &buf.exp[2]);  
 }  
 fprintf(fp, "%s\n", new.firstName);  
 fprintf(fp, "Receipt date: %d %d %d\n", new.res[0], new.res[1], new.res[2]);  
 fprintf(fp, "Date of expulsion: %d %d %d\n", new.exp[0], new.exp[1], new.exp[2]);  
 fclose(fp);  
}  
  
struct table \*File\_read(int \*flag, int \*len) {  
 struct table \*new = (struct table \*) malloc(2 \* sizeof(struct table));  
 FILE \*fp;  
 if ((fp = fopen("read.txt", "r")) == NULL) {  
 printf("Can't open file 'read.txt'\n");  
 \*flag = 1;  
 }  
 while (fscanf(fp, "%s", new[\*len].firstName) == 1) {  
 fscanf(fp, "%\*s%\*s%d %d %d", &new[\*len].res[0], &new[\*len].res[1], &new[\*len].res[2]);  
 fscanf(fp, "%\*s%\*s%\*s%d %d %d\n", &new[\*len].exp[0], &new[\*len].exp[1], &new[\*len].exp[2]);  
 new = (struct table \*) realloc(new, (\*len + 2) \* sizeof(struct table[2]));  
 (\*len)++;  
 }  
 fclose(fp);  
 for (int i = 0; i < \*len; ++i) input\_validation(new[i], flag);  
 return new;  
}  
  
int Find\_int(struct table \*new, int count, int value) {  
 int delta[2] = {INT\_MAX, 0};  
 for (int i = 0; i < count; ++i) {  
 for (int j = 0; j < 3; ++j) {  
 if (new[i].res[j] == value) return new[i].res[j];  
 if (new[i].exp[j] == value) return new[i].exp[j];  
 if (abs(value - new[i].res[j]) < delta[0]) {  
 delta[0] = abs(value - new[i].res[j]);  
 delta[1] = i;  
 }  
 if (abs(value - new[i].exp[j]) < delta[0]) {  
 delta[0] = abs(value - new[i].exp[j]);  
 delta[1] = i;  
 }  
 }  
 }  
 for (int i = 0; i < count; ++i) {  
 for (int j = 0; j < 3; ++j) {  
 if (abs(value - new[delta[1]].exp[j]) == delta[0]) {  
 printf("%s\nExpulsion date = %d\n", new[delta[1]].firstName, new[delta[1]].exp[j]);  
 return 0;  
 }  
 if (abs(value - new[delta[1]].res[j]) == delta[0]) {  
 printf("%s\nReceipt date = %d\n", new[delta[1]].firstName, new[delta[1]].res[j]);  
 return 0;  
 }  
 }  
 }  
 return 0;  
}  
  
int Find\_char(struct table \*new, int count, char \*value) {  
 int match, rez = 0;  
 for (int i = 0; i < count; ++i) {  
 match = 0;  
 for (int j = 0; j < strlen(value); ++j) {  
 if (new[i].firstName[0] == value[j]) match++;  
 }  
 if (match > rez) rez = match;  
 }  
 for (int i = 0; i < count; ++i) {  
 match = 0;  
 for (int j = 0; j < strlen(value); ++j) {  
 if (new[i].firstName[0] == value[j]) match++;  
 }  
 if (match == rez) {  
 printf("%s\n", new[i].firstName);  
 return 0;  
 }  
 }  
 return 0;  
}  
  
void Sort(struct table \*new, int count, int select) {  
 int i, j;  
 struct table temp;  
 for (i = 0; i < count - 1; i++)  
 for (j = 0; j < count - i - 1; j++) {  
 for (int k = 0; k < 3; ++k) {  
 switch (select) {  
 case 1:  
 if ((int) new[j].firstName[0] > (int) new[j + 1].firstName[0]) {  
 temp = new[j];  
 new[j] = new[j + 1];  
 new[j + 1] = temp;  
  
 }  
 break;  
 case 2:  
 if (new[j].res[k] > new[j + 1].res[k]) {  
 temp = new[j];  
 new[j] = new[j + 1];  
 new[j + 1] = temp;  
  
 }  
 break;  
 case 3:  
 if (new[j].exp[k] > new[j + 1].exp[k]) {  
 temp = new[j];  
 new[j] = new[j + 1];  
 new[j + 1] = temp;  
  
 }  
 break;  
 default:  
 printf("Error\n");  
 break;  
 }  
 }  
 }  
 for (int k = 0; k < count; ++k)  
 File\_write(new[k], 0, k, "sorted.txt");  
}  
  
void Delete\_value(struct table \*new, const int \*count) {  
 if (count[1] == 1) strcpy(new[count[0]].firstName, "");  
 if (count[1] == 2) new[count[0]].res[count[2]] = 0;  
 if (count[1] == 3) new[count[0]].exp[count[2]] = 0;  
  
}  
  
struct table \*Delete\_object(struct table \*new, const int \*count, int \*len) {  
 for (int i = \*count; i < \*len; ++i) {  
 printf("%s<->", new[i].firstName);  
 new[i] = new[i + 1];  
 printf("%s\n", new[i].firstName);  
 }  
 (\*len)--;  
 new = (struct table \*) realloc(new, \*len \* sizeof(struct table));  
 return new;  
}  
  
int Edit\_value(struct table \*new, const int \*count, int iValue, char \*cValue) {  
 if (count[1] == 1) strcpy(new[count[0]].firstName, cValue);  
 if (count[1] == 2) new[count[0]].res[count[2]] = iValue;  
 if (count[1] == 3) new[count[0]].exp[count[2]] = iValue;  
 return 0;  
}  
  
int Interface(struct table \*a, int len, int \*select\_id, int bool, int \*iValue, char \*cValue) {  
 for (int i = 0; i < len; ++i) {  
 printf("#%d - %s\n", i + 1, a[i].firstName);  
 }  
 printf("Chose user number: ");  
 scanf("%d", &select\_id[0]);  
 select\_id[0]--;  
 if ((select\_id[0] < 1) || (select\_id[0] > len + 1)) {  
 printf("Unavailable input\n");  
 return 1;  
 }  
 printf("#1 %s\n"  
 "#2 %d %d %d\n"  
 "#3 %d %d %d\n", a[select\_id[0]].firstName, a[select\_id[0]].res[0], a[select\_id[0]].res[1],  
 a[select\_id[0]].res[2], a[select\_id[0]].exp[0], a[select\_id[0]].exp[1], a[select\_id[0]].exp[2]);  
  
 printf("Select the field: ");  
 scanf("%d", &select\_id[1]);  
 if ((select\_id[1] < 1) || (select\_id[1] > 3)) {  
 printf("Unavailable input\n");  
 return 1;  
 }  
 if (select\_id[1] == 1) {  
 Delete\_value(a, select\_id);  
 if (bool == 1) {  
 printf("Enter a value: ");  
 scanf("%s", cValue);  
 }  
 } else {  
 printf("Select cell\n#1 dd\n#2 mm\n#3 yy\n");  
 scanf("%d", &select\_id[2]);  
 select\_id[2]--;  
 if ((select\_id[2] < 1) || (select\_id[2] > 3)) {  
 printf("Unavailable input\n");  
 return 1;  
 }  
 if (bool == 1) {  
 printf("Enter a value: ");  
 scanf("%d", iValue);  
 }  
 }  
 return 0;  
}  
  
void Average(struct table \*new, int len) {  
 double delta[3];  
 for (int j = 0; j < 3; ++j) delta[j] = 0;  
 for (int i = 0; i < len; ++i)  
 for (int j = 0; j < 3; ++j)  
 delta[j] += abs(new[i].res[j] - new[i].exp[j]);  
 for (int j = 0; j < 3; ++j) {  
 delta[j] /= len;  
 printf("%.2f ", round(delta[j] \* 1000) / 1000);  
 }  
 printf("\n");  
}  
  
int check(int select, int a, int b) {  
 if (select < a || select > b) {  
 printf("Unavailable input\n");  
 return 1;  
 } else return 0;  
}  
  
int Ask() {  
 int flag = 0, select, fInt, len = 0, rez, iValue, buf= 0;  
 int select\_id[3];  
 char \*cValue = (char \*) malloc(20 \* sizeof(char));  
 char fChar[20];  
 struct table \*a;  
 while (1) {  
 printf("1 <->Enter table from console?\n"  
 "2 <->Read table from file?\n"  
 "3 <->Show tables?\n"  
 "4 <->Write tables in file?\n"  
 "5 <->Sort table?\n"  
 "6 <->Find value in table?\n"  
 "7 <->Delete value?\n"  
 "8 <->Delete object?\n"  
 "9 <->Edit value?\n"  
 "10<->Show the average length of study?\n"  
 "11<->Exit\n");  
 scanf("%d", &select);  
 if (check(select, 1, 12)) return 0;  
 if ((len == 0) && ((select == 3) || (select > 4) && (select < 11))) {  
 printf("\tVoid table. Add objects!\n\n");  
 continue;  
 }  
 switch (select) {  
 case 1:  
 printf("How many table do you want to enter?\n");  
 scanf("%d", &buf);  
 len+=buf;  
 a = (struct table \*) realloc(a, sizeof(struct table) \* len);  
 for (int i = len - buf; i < len; ++i) {  
 a[i] = Input(&flag);  
 if (flag == 1) return 0;  
 }  
 break;  
 case 2:  
 a = File\_read(&flag, &len);  
 printf("Successful!\n");  
 break;  
 case 3:  
 for (int i = 0; i < len; ++i) Output(a[i]);  
 break;  
 case 4:  
 for (int i = 0; i < len; ++i) File\_write(a[i], &flag, i, "write.txt");  
 break;  
 case 5:  
 printf("Select sort:\n"  
 "1-Sort by last name\n"  
 "2-Sort by date of receipt\n"  
 "3-Sort by expiration date\n");  
 scanf("%d", &select);  
 if (check(select, 1, 3)) return 0;  
 Sort(a, len, select);  
 for (int i = 0; i < len; ++i) {  
 printf("%s\n", a[i].firstName);  
 switch (select) {  
 case 1:  
 continue;  
 case 2:  
 printf("%d %d %d\n", a[i].res[0], a[i].res[1], a[i].res[2]);  
 break;  
 case 3:  
 printf("%d %d %d\n", a[i].exp[0], a[i].exp[1], a[i].exp[2]);  
 break;  
 default:  
 return 0;  
 }  
 }  
 break;  
 case 6:  
 printf("What do you want to find int or char?\n"  
 "1-Int\n"  
 "2-Char\n");  
 scanf("%d", &select);  
 if (check(select, 1, 2)) return 0;  
 printf("Enter value\n");  
 switch (select) {  
 case 1:  
 scanf("%d", &fInt);  
 Find\_int(a, len, fInt);  
 break;  
 case 2:  
 scanf("%s", fChar);  
 Find\_char(a, len, fChar);  
 break;  
 default:  
 printf("Unavailable input\n");  
 return 0;  
 }  
 break;  
 case 7:  
 rez = Interface(a, len, select\_id, 0, 0, 0);  
 if (rez == 1) return 0;  
 if (rez == 2) Delete\_value(a, select\_id);  
 printf("#1 %s\n"  
 "#2 %d %d %d\n"  
 "#3 %d %d %d\n", a[select\_id[0]].firstName, a[select\_id[0]].res[0], a[select\_id[0]].res[1],  
 a[select\_id[0]].res[2], a[select\_id[0]].exp[0], a[select\_id[0]].exp[1],  
 a[select\_id[0]].exp[2]);  
 break;  
 case 8:  
 for (int i = 0; i < len; ++i) {  
 printf("####%d####\n", i);  
 printf("%s\n", a[i].firstName);  
 }  
 printf("Enter the object number to be deleted\n");  
 scanf("%d", &select\_id[2]);  
 if (check(select\_id[2], 1, len)) return 0;  
 a = Delete\_object(a, &select\_id[2], &len);  
 for (int i = 0; i < len; ++i) {  
 printf("####%d####\n", i);  
 printf("%s\n", a[i].firstName);  
 }  
 break;  
 case 9:  
 rez = Interface(a, len, select\_id, 1, &iValue, cValue);  
 if (rez == 1) return 0;  
 Edit\_value(a, select\_id, iValue, cValue);  
 printf("\n%s\n"  
 "%d %d %d\n"  
 "%d %d %d\n", a[select\_id[0]].firstName, a[select\_id[0]].res[0], a[select\_id[0]].res[1],  
 a[select\_id[0]].res[2], a[select\_id[0]].exp[0], a[select\_id[0]].exp[1],  
 a[select\_id[0]].exp[2]);  
 break;  
 case 10:  
 printf(" dd mm yy\n");  
 Average(a, len);  
 break;  
 case 11:  
 return 0;  
 default:  
 printf("Unavailable input\n");  
 }  
 }  
}  
  
int main() {  
 if (!Ask()) return 0;  
}