Міністерство освіти і науки України

Дніпропетровський національний університет

імені Олеся Гончара

ФФЕКС

Кафедра ЕОМ

**Звіз з лабораторної роботи № 3**

Виконав: студент 2 курсу

Шейко Р.О.

Науковий керівник:

Викладач кафедри EOM

Спірінцева Ольга Володимирівна

м. Дніпро

2022

Завдання

1.Визначити функції запису та зчитування структури для двійкового та текстового файлів. У всіх функцій одним із аргументів має бути посилання на структурний тип, у функцій роботи з двійковими файлами ще одним аргументом має бути покажчик на тип FILE.

Використовувати структурний тип та функції введення та виведення на екран даних, визначені у програмі до лабораторної роботи №2.

2. Визначити глобальні покажчики на тип FILE для функцій роботи з двійковими файлами.

3. Визначити функцію main(), у якій створити:

+ покажчики на тип FILE для функцій роботи з текстовими файлами;

+ покажчик на структурний тип для організації динамічного масиву структур;

+ перший динамічний масив структур невеликого розміру (3 – 4 елементи) – для введення даних та їх збереження у файлах;

+ другий динамічний масив структур такого ж розміру для зчитування даних із текстового файлу;

+ Третій динамічний масив структур – для зчитування даних із двійкового файлу.

4. Ввести дані до елементів першого динамічного масиву за допомогою функцій, розроблених у лабораторній роботі №2.

5. Зберегти вміст елементів масиву структур у текстовому та двійковому файлах.

6 Вважати вміст текстового файлу в елементах другого масиву та вивести на екран. Порівняти вміст першого та другого масивів.

7. Вважати вміст двійкового файлу в елементи третього масиву та вивести на екран. Порівняти вміст першого та третього масивів.

8. Переглянути вміст двійкового та текстового файлів за допомогою будь-якого тестового редактора або програми переглядача (для перегляду тестового файлу можна використовувати «Блокнот» Windows, для перегляду двійкового файлу рекомендується Total Commander або аналогічна програма). Встановити відповідність між елементами даних у першому масиві та текстовому файлі. Спробувати зробити те саме для двійкового файлу. Зберегти файл з тестом програми для наступних робіт.

Хід роботи

1. По перше, у цій роботі відсутнє кодування у двійковий код. Чому? Вважаю непотрібною у житті інформацією. Краще б SOLID вивчили на прикладі .NET. У професійній середі дуже потрібна штука.

По-друге. Я вважаю архітектурно невиправданим використання великої кількості структур. Моє рішення – манюпулювати однією структурою за для того, щоб зберегти пам’ять та в одному тестуванні використовувати і зчитування з файлу, й заповнення від руки.

Ще. Використання глобальних змінних(або змінних, що довгий час зберігаються у пам’яті) я вважаю дуже НЕБЕЗПЕЧНИМ, тому що можна викрасти ці данні, підмінити їх при, наприклад, у трьох однакових пакетах з п’яти, й, наприклад, зробити собі дуже велику кількість грошей.

1. Для роботи з двійковими файлами ще одним аргументом має бути покажчик на тип FILE:

FILE\* data2;//строчка 6

Визначити функції запису та зчитування структури для двійкового та текстового файлів.

//строчка 141

List initKnigaFileR(FILE\* data) {

List list;

fscanf\_s(data, "%s ", list.Name, N);

char SecondName[N];

fscanf\_s(data, "%s ", SecondName, N - 1);

list.SecondName = (char\*)malloc((strlen(SecondName) + 1) \* sizeof(char));

writeToFrom\_L2(list.SecondName, SecondName);

list.SecondName[(strlen(SecondName))] = '\0';

fscanf\_s(data, "%f ", &list.Number);

fscanf\_s(data, "%ld ", &list.SizeOfMassive);

long\* Massive = (long\*)malloc(list.SizeOfMassive \* sizeof(long));

for(int i = 0; i < list.SizeOfMassive; i++)

fscanf\_s(data, "%ld ", &Massive[i]);

list.Massive = Massive;

return list;

}

List\* initArrayFile(char fileName[], int dimension) {

List\* massive = (List\*)malloc(dimension \* sizeof(List));

if (massive == NULL)

return NULL;

FILE\* data;

fopen\_s(&data, fileName, "r");

if (!data) {

printf("Error");

exit(1);

}

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

massive[i] = initKnigaFileR(data);

fclose(data);

return massive;

}

//строчка 173

Функція initKnigaFileR зчитуває з файлу структуру й повертає її.

Функція initArrayFile відкриває файл й у циклі заповнює масив структур, що потім повертає до головної функції.

2). Визначити глобальні покажчики на тип FILE для функцій роботи з двійковими файлами.

FILE\* data2;//строчка 6

1. - покажчики на тип FILE для функцій роботи з текстовими файлами

Найбільш ефективний й безпечний спосіб ініціалюзувати в фунціях.

- покажчик на структурний тип для організації динамічного масиву структур;

List\*.

- перший динамічний масив структур невеликого розміру (3 – 4 елементи) – для введення даних та їх збереження у файлах;

List\* massive = (List\*)malloc(dimension \* sizeof(List));

-другий динамічний масив структур такого ж розміру для зчитування даних із текстового файлу;

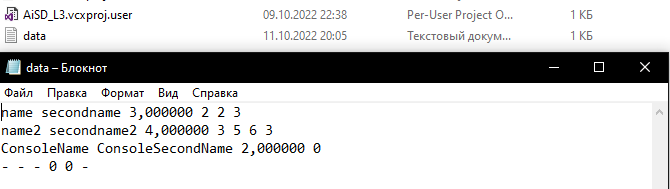
-третій динамічний масив структур – для зчитування даних із двійкового файлу

Другий, як і третій, не додавав, тому що можна у програмі одним масивом по різному маніпулювати.

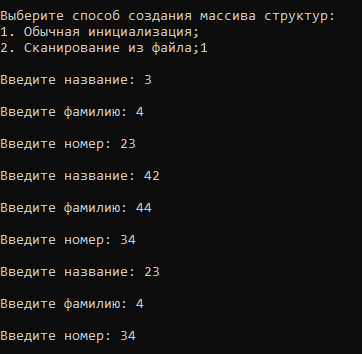
1. Ввести дані до елементів першого динамічного масиву за допомогою функцій, розроблених у лабораторній роботі №2.

Зроблено. Перевірити можна у лістингу усієї програми.

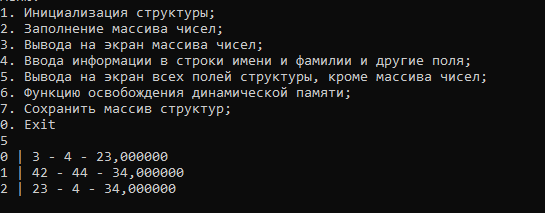
1. Зберегти вміст елементів масиву структур у текстовому та двійковому файлах.



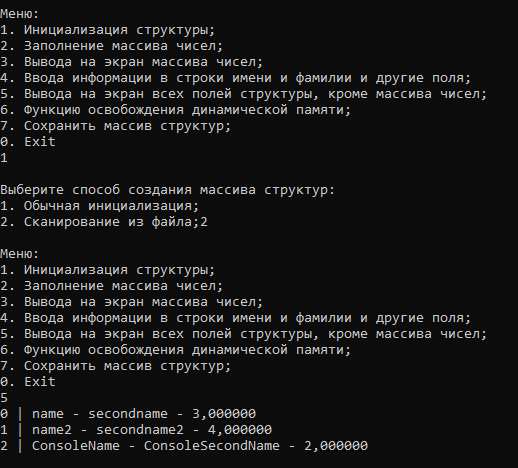
1. Вважати вміст текстового файлу в елементах другого масиву та вивести на екран. Порівняти вміст першого та другого масивів.



Ініціалізація структури деф способом



Output деф спосіб



Ініціалізація через файл

6 Вважати вміст текстового файлу в елементах другого масиву та вивести на екран. Порівняти вміст першого та другого масивів.

**Тестування**

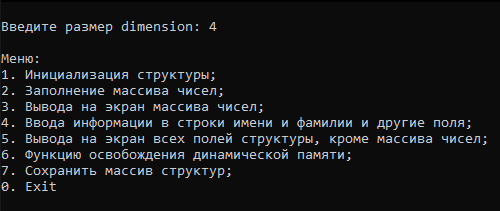


Рис 1. Початок роботи

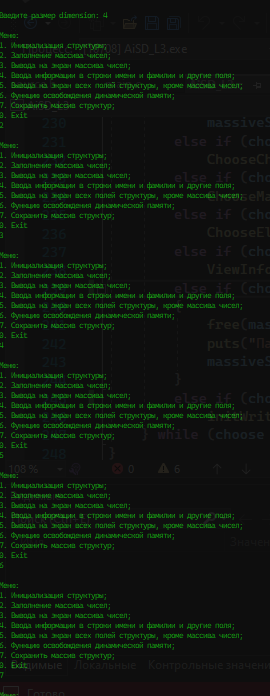


Рис 2. Спроба зробити шось з не ініційованною консолю

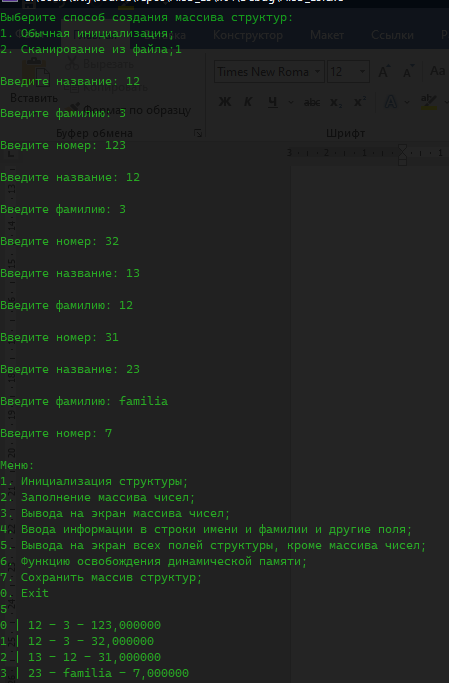


Рис 3. Def Ініцюювання структури через Console

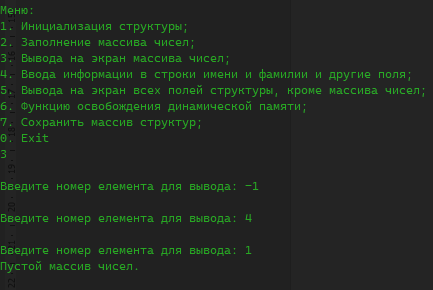


Рис 4. Тестування та спроба вивести нестворенний масив

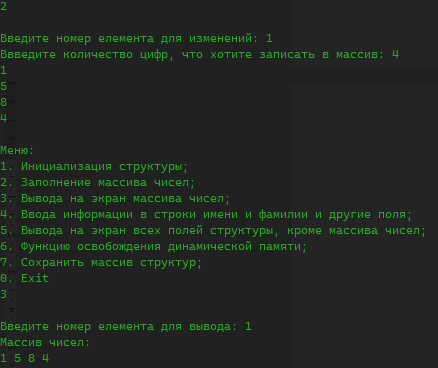


Рис 5. Заповнення масиву цифр в вибранній структурі

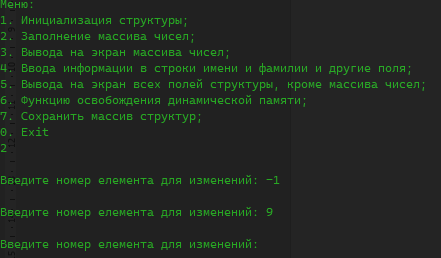


Рис 6. Тестування вибору неіснуючого масиву



Рис 7. Спроба записати неможливу кількість цифр

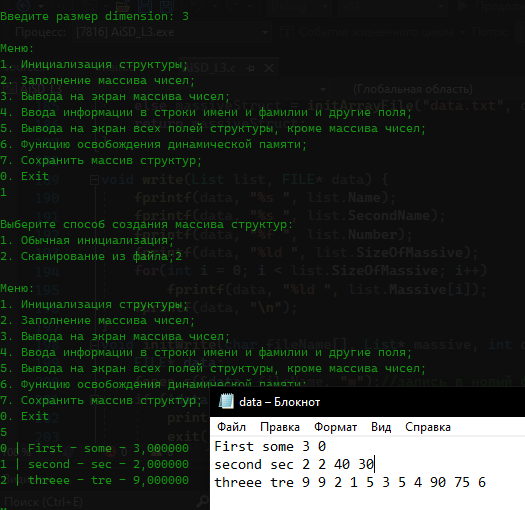


Рис 8. Ініціалізація через файл.

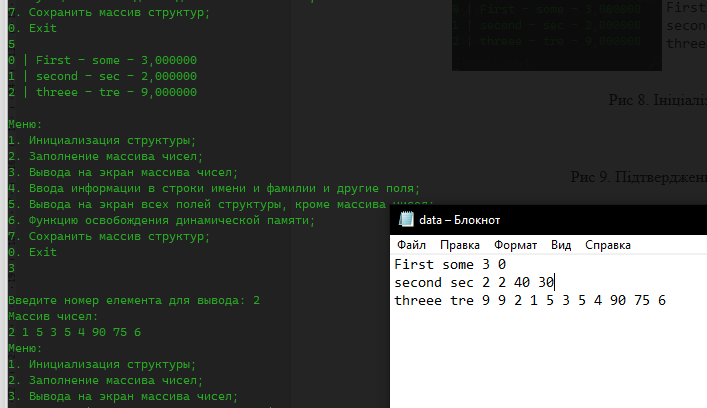


Рис 9. Підтвердження зчитування та рис 8

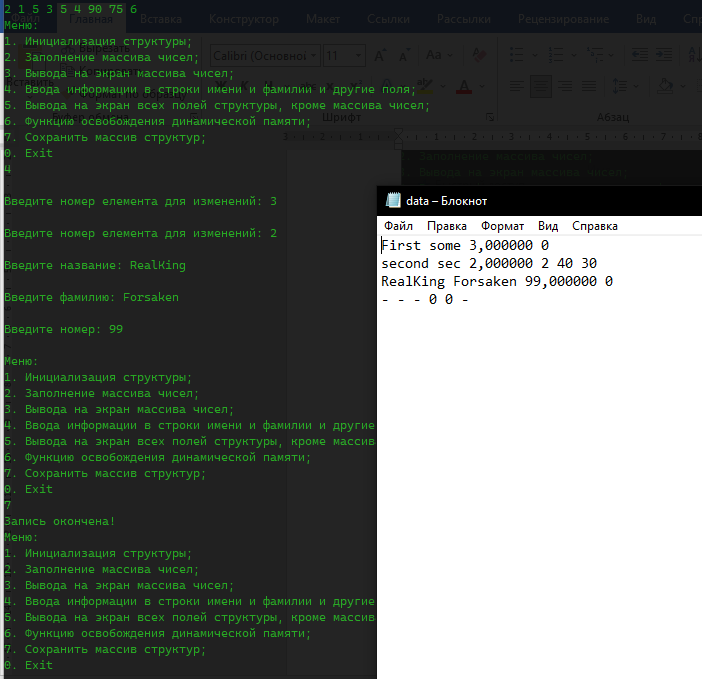


Рис 10. Ілюстрація запису.

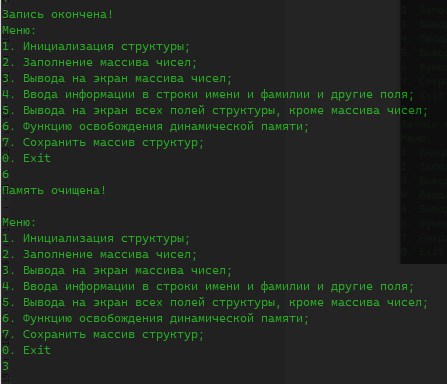


Рис 11. Очищення пам’яті та спроба маніпулювання структурою

Лістинг з лабораторної роботи. Файл <main.c>

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <locale.h>

#define N 50

FILE\* data2;

typedef struct {

char Name[N];

char\* SecondName;

float Number;//nomer//Переменная плаваю- щего типа.

unsigned long SizeOfMassive;//Переменная беззнакового длинного целого типа;

long\* Massive;//Указатель на длинный целый тип;

} List;

void writeToFrom\_L2(char a[], char b[]) {

for (int i = 0; i < strlen(b) + 1; i++)

a[i] = b[i];

}

void safeScanf\_L2(int\* a) {

char temp[N];

int error = 0;

do {

error = 0;

scanf\_s("%s", &temp, N - 1);

for (int a = 0; a < strlen(temp); a++)

if (atoi(temp) == 0 && !(('0' <= temp[a]) && (temp[a] <= '9')))

error = 1;

if (error == 1)

printf("Вводите только числа!\n");

else {

\*a = atoi(temp);

break;

}

} while (1);

}

List TakeElement\_L2() {

List list;

printf("\nВведите название: ");

scanf\_s("%s", &list.Name, N);

char SName[N];

printf("\nВведите фамилию: ");

scanf\_s("%s", &SName, N - 1);

list.SecondName = (char\*)malloc((strlen(SName) + 1) \* sizeof(char));

writeToFrom\_L2(list.SecondName, SName);

printf("\nВведите номер: ");

scanf\_s("%f", &list.Number);

list.SizeOfMassive = 0;

return list;

}

List\* initArray\_L2(int dimension) {

List\* massive = (List\*)malloc(dimension \* sizeof(List));

if (massive == NULL)

return NULL;

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

massive[i] = TakeElement\_L2();

return massive;

}

void ViewInfoElement\_L2(int i, List list) {

printf("%d | %s - %s - %f\n", i, list.Name, list.SecondName, list.Number);

}

void ViewInfo\_L2(List\* lists, int dimension) {

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

ViewInfoElement\_L2(i, lists[i]);

}

List Change\_L2(List list\_) {

List list;

int choose = 0;

do {

printf("Ввведите количество цифр, что хотите записать в массив: ");

safeScanf\_L2(&choose);

if (choose >= 1)

break;

} while (1);

list.SizeOfMassive = (long)choose;

list.Massive = (long\*)malloc(list.SizeOfMassive \* sizeof(long));

for (int i = 0, num = 0; i < list.SizeOfMassive; i++) {

safeScanf\_L2(&num);

list.Massive[i] = num;

}

return list;

}

void ChooseChangeMassiveInLists\_L2(List\* lists, int dimension) {

int choosed = 0;

do {

printf("\nВведите номер елемента для изменений: ");

safeScanf\_L2(&choosed);

if (choosed >= 0 && choosed < dimension)

break;

} while (1);

lists[choosed] = Change\_L2(lists[choosed]);

}

void MassiveOutPut\_L2(List list) {

puts("Массив чисел: ");

for (int i = 0; i < list.SizeOfMassive; i++)

printf("%ld ", list.Massive[i]);

}

void ChooseMassiveOutPut\_L2(List\* lists, int dimension) {

int choosed = 0;

do {

printf("\nВведите номер елемента для вывода: ");

safeScanf\_L2(&choosed);

if (choosed >= 0 && choosed < dimension)

break;

} while (1);

if (lists[choosed].SizeOfMassive > 0)

MassiveOutPut\_L2(lists[choosed]);

else puts("Пустой массив чисел.");

}

void ChooseElementToChange\_L2(List\* lists, int dimension) {

int choosed = 0;

do {

printf("\nВведите номер елемента для изменений: ");

safeScanf\_L2(&choosed);

if (choosed >= 0 && choosed < dimension)

break;

} while (1);

lists[choosed] = TakeElement\_L2();

}

int TakeDimension\_L2() {

int chooseNum = 0;

do {

printf("\nВведите размер dimension: ");

safeScanf\_L2(&chooseNum);

if (chooseNum > 0 && chooseNum < 10)

break;

else if (chooseNum >= 10)

printf("\nНе нужно делать больше 10, неудобно потом будет)");

} while (1);

return chooseNum;

}

List initKnigaFileR(FILE\* data) {

List list;

fscanf\_s(data, "%s ", list.Name, N);

char SecondName[N];

fscanf\_s(data, "%s ", SecondName, N - 1);

list.SecondName = (char\*)malloc((strlen(SecondName) + 1) \* sizeof(char));

writeToFrom\_L2(list.SecondName, SecondName);

list.SecondName[(strlen(SecondName))] = '\0';

fscanf\_s(data, "%f ", &list.Number);

fscanf\_s(data, "%ld ", &list.SizeOfMassive);

long\* Massive = (long\*)malloc(list.SizeOfMassive \* sizeof(long));

for(int i = 0; i < list.SizeOfMassive; i++)

fscanf\_s(data, "%ld ", &Massive[i]);

list.Massive = Massive;

return list;

}

List\* initArrayFile(char fileName[], int dimension) {

List\* massive = (List\*)malloc(dimension \* sizeof(List));

if (massive == NULL)

return NULL;

FILE\* data;

fopen\_s(&data, fileName, "r");

if (!data) {

printf("Error");

exit(1);

}

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

massive[i] = initKnigaFileR(data);

fclose(data);

return massive;

}

List\* ChooseOfInitMassiveStruct(List\* massiveStruct, int dimension) {

int choosed = 0;

do {

printf("\nВыберите способ создания массива структур: \

\n1. Обычная инициализация;\

\n2. Сканирование из файла;");

safeScanf\_L2(&choosed);

} while (choosed != 1 && choosed != 2);

if (choosed == 1)

massiveStruct = initArray\_L2(dimension);

else massiveStruct = initArrayFile("data.txt", dimension);

return massiveStruct;

}

void write(List list, FILE\* data) {

fprintf(data, "%s ", list.Name);

fprintf(data, "%s ", list.SecondName);

fprintf(data, "%f ", list.Number);

fprintf(data, "%ld ", list.SizeOfMassive);

for(int i = 0; i < list.SizeOfMassive; i++)

fprintf(data, "%ld ", list.Massive[i]);

fprintf(data, "\n");

}

void initWrite(char fileName[], List\* massive, int dimension) {

FILE\* data;

fopen\_s(&data, fileName, "w");//запись в новый файл

if (!data) {

printf("Error");

exit(1);

}

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

write(massive[i], data);

fprintf(data, "- - - 0 0 -\n");

fclose(data);

printf("Запись окончена!");

}

void main()

{

setlocale(LC\_ALL, "Rus");

List\* massiveStruct = NULL;

int choose = 0;

int dimension = TakeDimension\_L2();

do {

puts("\nМеню: \

\n1. Инициализация структуры;\

\n2. Заполнение массива чисел;\

\n3. Вывода на экран массива чисел;\

\n4. Ввода информации в строки имени и фамилии и другие поля;\

\n5. Bывода на экран всех полей структуры, кроме массива чиcел;\

\n6. Функцию освобождения динамической памяти;\

\n7. Сохранить массив структур;\

\n0. Exit");

safeScanf\_L2(&choose);

if (choose == 1)

massiveStruct = ChooseOfInitMassiveStruct(massiveStruct, dimension);

else if (choose == 2 && massiveStruct != NULL)

ChooseChangeMassiveInLists\_L2(massiveStruct, dimension);

else if (choose == 3 && massiveStruct != NULL)

ChooseMassiveOutPut\_L2(massiveStruct, dimension);

else if (choose == 4 && massiveStruct != NULL)

ChooseElementToChange\_L2(massiveStruct, dimension);

else if (choose == 5 && massiveStruct != NULL)

ViewInfo\_L2(massiveStruct, dimension);

else if (choose == 6)

{

free(massiveStruct);//очищення пам'яті

puts("Память очищена!");

massiveStruct = NULL;

}

else if (choose == 7 && massiveStruct != NULL)

initWrite("data.txt", massiveStruct, dimension);

} while (choose != 0);

}