***Министерство образования Республики Беларусь***

***Учреждение Образования***

***«Брестский Государственный Технический Университет»***

***Кафедра ИИТ***

**Лабораторная работа №11**

**По дисциплине ОАиП за II семестр**

**Тема: «Структуры, массивы, указатели.**

**Динамические массивы структур.**

**Функции»**

**Выполнил:**

Студент 1-го курса

Группы ИИ-15(1)

Волк И. А.

**Проверила:**

Хацкевич М. В.

Брест 2018

Цель: научиться работать с указателями на функции. Улучшить умения работы с динамическими массивами структур и функциями.

1.Осуществлять запуск исполняемого файла с командной строки с одним параметром командной строки.

2.Реализовать меню, разработанное с применением массива указателей на функции.

**Для выбранной базы данных написать следующие функции:**

1. Функцию формирования динамического одномерного массива структур, значения которых вводятся с клавиатуры. Предусмотреть возможность заполнения одного поля структуры, используя известные значения других полей структуры. При вводе структур можно реализовать один из следующих механизмов:

• ввод заранее заданного количества структур;

• ввод до появления структуры с заданным признаком;

• диалог с пользователем о необходимости продолжать ввод.

2. Функцию просмотра содержимого динамического массива структур.

3. Функцию дополнения уже существующего массива структур новыми структурами. Данная функция должна содержать параметр со значение по умолчанию (количество новых структур).

Пример,

void func (int a = 0, int b = 10){}

//вызов

func();

func(-1);

func(-1, 99);

4. Функцию поиска и вывода на экран структуры (структур) с заданным значением элемента.

5. Функцию упорядочения массива структур по заданному полю (элементу).

6.Организовать функцию с переменным числом параметров.

Код программы:

#include "stdafx.h"

#include <Windows.h>

#include <stdio.h>

#include <conio.h>

#include <iostream>

#include <string>

using namespace std;

struct fullNameStruct

{

char name[1024];

char lastName[1024];

char patronymic[1024];

};

struct dateOfBirthStruct

{

int year, month, date;

};

struct homeAddressStruct

{

char postcode[1024];

char country[1024];

char region[1024];

char district[1024];

char town[1024];

char street[1024];

char house[1024];

char apartmentNumber[1024];

};

struct highSchoolStruct

{

char name[1024];

char course[1024];

char group[1024];

int averageGrade;

char specialty[1024];

};

struct student

{

int numOfStud;

fullNameStruct fullName;

enum

{

male = 1,

female

}gender;

char nationality[1024];

double hight, wight;

dateOfBirthStruct dateOfBirth;

char phoneNumber[1024];

homeAddressStruct address;

highSchoolStruct highSchool;

};

bool gender(student\*, int, int);

bool yearOfBirth(student\*, int, int);

bool lastName(student\*, int, char[1024]);

bool country(student\*, int, char[1024]);

bool town(student\*, int, char[1024]);

bool highSchoolName(student\*, int, char[1024]);

student\* biggerArray(student\*, int\*);

void searchResult(student\*, int\*, char[1024], bool(\*)(student\*, int, char[1024]));

void searchResult(student\*, int\*, int, bool(\*)(student\*, int, int));

void sort(student\*, int\*, int);

/\* Main functions \*/

int menu();

void outputStruct(student);

student\* inputStructArray(student\*, int\*);

student\* outputStructArray(student\*, int\*);

student\* addStructArray(student\*, int\*, int);

student\* searchStruct(student\*, int\*);

student\* sortStructArray(student\*, int\*);

student\* finishProgram(student\*, int\*);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

student\* (\*functionArray[])(student\*, int\*) = { inputStructArray,

outputStructArray,

searchStruct,

sortStructArray,

finishProgram };

bool(\*functionForSearchInt[])(student\*, int, int) = { gender,

yearOfBirth };

bool(\*functionForSearchString[])(student\*, int, char[1024]) = { lastName,

country,

town,

highSchoolName};

int main(int argc, char \*argv[])

{

char password[] = "password";

if (argc != 2 || strcmp(argv[1], password))

{

cout << "Login failed!";

cout << endl;

system("pause");

return 0;

}

student \* arrayOfStruct = new student;

int sizeOfArrayOfStruct = 0;

while (1)

{

int menuReturn = menu();

if (menuReturn < 2)

arrayOfStruct = (functionArray[menuReturn])(arrayOfStruct, &sizeOfArrayOfStruct);

else if (menuReturn == 2)

arrayOfStruct = addStructArray(arrayOfStruct, &sizeOfArrayOfStruct, 1);

else if (menuReturn > 2)

arrayOfStruct = (functionArray[menuReturn - 1])(arrayOfStruct, &sizeOfArrayOfStruct);

}

system("pause");

return 0;

}

int menu()

{

int answer;

while (1)

{

system("cls");

cout << "Menu: " << endl;

cout << "1. Input" << endl;

cout << "2. Output" << endl;

cout << "3. Add" << endl;

cout << "4. Search" << endl;

cout << "5. Sort" << endl;

cout << "6. Exit" << endl;

answer = \_getch();

if (answer < 49 || answer > 54)

{

system("cls");

cout << "Error!";

Sleep(1000);

continue;

}

else

break;

}

return answer-49;

}

student\* inputStructArray(student\* arrayOfStruct, int \*sizeOfArrayOfStruct)

{

\*sizeOfArrayOfStruct = 0;

student \* arrayOfStructResult;

while (1)

{

cin.clear();

int sizeOfArrayOfStructResult = \*sizeOfArrayOfStruct;

system("cls");

arrayOfStructResult = biggerArray(arrayOfStruct, &sizeOfArrayOfStructResult);

int numOfElement = sizeOfArrayOfStructResult - 1;

arrayOfStructResult[numOfElement].numOfStud = sizeOfArrayOfStructResult;

cout << "Element #" << arrayOfStructResult[numOfElement].numOfStud << endl;

/\*Inputing\*/

cout << "Name: "; cin >> arrayOfStructResult[numOfElement].fullName.name;

cout << "Last name: "; cin >> arrayOfStructResult[numOfElement].fullName.lastName;

cout << "Patronymic: "; cin >> arrayOfStructResult[numOfElement].fullName.patronymic;

int gender;

cout << "Gender(1 - male, 2 - female): "; cin >> gender;

(gender==1)?

(arrayOfStructResult[numOfElement].gender = student::male)

:(arrayOfStructResult[numOfElement].gender = student::female);

cout << "Nationality: "; cin >> arrayOfStructResult[numOfElement].nationality;

cout << "Hight: "; cin >> arrayOfStructResult[numOfElement].hight;

cout << "Wight: "; cin >> arrayOfStructResult[numOfElement].wight;

cout << "Year of birth: "; cin >> arrayOfStructResult[numOfElement].dateOfBirth.year;

cout << "Month of birth: "; cin >> arrayOfStructResult[numOfElement].dateOfBirth.month;

cout << "Date of birth: "; cin >> arrayOfStructResult[numOfElement].dateOfBirth.date;

cout << "Phone number: "; cin >> arrayOfStructResult[numOfElement].phoneNumber;

cout << "Postcode: "; cin >> arrayOfStructResult[numOfElement].address.postcode;

cout << "Country: "; cin >> arrayOfStructResult[numOfElement].address.country;

cout << "Region: "; cin >> arrayOfStructResult[numOfElement].address.region;

cout << "District: "; cin >> arrayOfStructResult[numOfElement].address.district;

cout << "Town: "; cin >> arrayOfStructResult[numOfElement].address.town;

cout << "Street: "; cin >> arrayOfStructResult[numOfElement].address.street;

cout << "House: "; cin >> arrayOfStructResult[numOfElement].address.house;

cout << "Apartment number: "; cin >> arrayOfStructResult[numOfElement].address.apartmentNumber;

cout << "#High school#\nName: "; cin >> arrayOfStructResult[numOfElement].highSchool.name;

cout << "Course: "; cin >> arrayOfStructResult[numOfElement].highSchool.course;

cout << "Group: "; cin >> arrayOfStructResult[numOfElement].highSchool.group;

cout << "Average grade: "; cin >> arrayOfStructResult[numOfElement].highSchool.averageGrade;

cout << "Specialty: "; cin >> arrayOfStructResult[numOfElement].highSchool.specialty;

/\*\*\*\*\*\*\*\*\*\*/

arrayOfStruct = biggerArray(arrayOfStruct, sizeOfArrayOfStruct);

for(int i = 0; i < \*sizeOfArrayOfStruct; i++)

arrayOfStruct[i] = arrayOfStructResult[i];

system("cls");

bool answer;

cout << "Do you want to input more?(0 - no)\n Answer: " << endl;

cin >> answer;

if (!answer)

break;

}

system("cls");

return arrayOfStruct;

}

void outputStruct(student stud)

{

cout << "Struct #" << stud.numOfStud << endl;

cout << "Name: " << stud.fullName.name << endl;

cout << "Last name: " << stud.fullName.lastName << endl;

cout << "Patronymic: " << stud.fullName.patronymic << endl;

cout << "Gender: " << stud.gender << endl;

cout << "Nationality: " << stud.nationality << endl;

cout << "Hight: " << stud.hight << endl;

cout << "Wight: " << stud.wight << endl;

cout << "Year of birth: " << stud.dateOfBirth.year << endl;

cout << "Month of birth: " << stud.dateOfBirth.month << endl;

cout << "Date of birth: " << stud.dateOfBirth.date << endl;

cout << "Phone number: " << stud.phoneNumber << endl;

cout << "Postcode: " << stud.address.postcode << endl;

cout << "Country: " << stud.address.country << endl;

cout << "Region: " << stud.address.region << endl;

cout << "District: " << stud.address.district << endl;

cout << "Town: " << stud.address.town << endl;

cout << "Street: " << stud.address.street << endl;

cout << "House: " << stud.address.house << endl;

cout << "Apartment number: " << stud.address.apartmentNumber << endl;

cout << "#High school#\nName: " << stud.highSchool.name << endl;

cout << "Course: " << stud.highSchool.course << endl;

cout << "Group: " << stud.highSchool.group << endl;

cout << "Average grade: " << stud.highSchool.averageGrade << endl;

cout << "Specialty: " << stud.highSchool.specialty;

}

student\* outputStructArray(student\* arrayOfStruct, int \*sizeOfArrayOfStruct)

{

for (int numOfElement = 0; numOfElement < \*sizeOfArrayOfStruct;)

{

system("cls");

outputStruct(arrayOfStruct[numOfElement]);

int ch;

while (1)

{

ch = \_getch();

if (ch == 75)

if (numOfElement != 0)

{

numOfElement--; break;

}

if (ch == 77)

if (numOfElement != (\*sizeOfArrayOfStruct - 1))

{

numOfElement++; break;

}

if (ch == 27)

return arrayOfStruct;

}

}

return arrayOfStruct;

}

student\* addStructArray(student\* arrayOfStruct, int \*sizeOfArrayOfStruct, int numOfNewStructs = 1)

{

student \* arrayOfStructResult;

for(int j = 0; j < numOfNewStructs; j++)

{

cin.clear();

int sizeOfArrayOfStructResult = \*sizeOfArrayOfStruct;

system("cls");

arrayOfStructResult = biggerArray(arrayOfStruct, &sizeOfArrayOfStructResult);

cout << "Element #" << sizeOfArrayOfStructResult << endl;

int numOfElement = sizeOfArrayOfStructResult - 1;

/\*Inputing\*/

cout << "Name: "; cin >> arrayOfStructResult[numOfElement].fullName.name;

cout << "Last name: "; cin >> arrayOfStructResult[numOfElement].fullName.lastName;

cout << "Patronymic: "; cin >> arrayOfStructResult[numOfElement].fullName.patronymic;

int gender;

cout << "Gender(1 - male, 2 - female): "; cin >> gender;

(gender == 1) ?

(arrayOfStructResult[numOfElement].gender = student::male)

: (arrayOfStructResult[numOfElement].gender = student::female);

cout << "Nationality: "; cin >> arrayOfStructResult[numOfElement].nationality;

cout << "Hight: "; cin >> arrayOfStructResult[numOfElement].hight;

cout << "Wight: "; cin >> arrayOfStructResult[numOfElement].wight;

cout << "Year of birth: "; cin >> arrayOfStructResult[numOfElement].dateOfBirth.year;

cout << "Month of birth: "; cin >> arrayOfStructResult[numOfElement].dateOfBirth.month;

cout << "Date of birth: "; cin >> arrayOfStructResult[numOfElement].dateOfBirth.date;

cout << "Phone number: "; cin >> arrayOfStructResult[numOfElement].phoneNumber;

cout << "Postcode: "; cin >> arrayOfStructResult[numOfElement].address.postcode;

cout << "Country: "; cin >> arrayOfStructResult[numOfElement].address.country;

cout << "Region: "; cin >> arrayOfStructResult[numOfElement].address.region;

cout << "District: "; cin >> arrayOfStructResult[numOfElement].address.district;

cout << "Town: "; cin >> arrayOfStructResult[numOfElement].address.town;

cout << "Street: "; cin >> arrayOfStructResult[numOfElement].address.street;

cout << "House: "; cin >> arrayOfStructResult[numOfElement].address.house;

cout << "Apartment number: "; cin >> arrayOfStructResult[numOfElement].address.apartmentNumber;

cout << "#High school#\nName: "; cin >> arrayOfStructResult[numOfElement].highSchool.name;

cout << "Course: "; cin >> arrayOfStructResult[numOfElement].highSchool.course;

cout << "Group: "; cin >> arrayOfStructResult[numOfElement].highSchool.group;

cout << "Average grade: "; cin >> arrayOfStructResult[numOfElement].highSchool.averageGrade;

cout << "Specialty: "; cin >> arrayOfStructResult[numOfElement].highSchool.specialty;

/\*\*\*\*\*\*\*\*\*\*/

arrayOfStruct = biggerArray(arrayOfStruct, sizeOfArrayOfStruct);

for (int i = 0; i < \*sizeOfArrayOfStruct; i++)

arrayOfStruct[i] = arrayOfStructResult[i];

}

system("cls");

return arrayOfStruct;

}

student\* searchStruct(student\* arrayOfStruct, int \*sizeOfArrayOfStruct)

{

while (1)

{

system("cls");

cout << "1. Last name " << endl;

cout << "2. Gender " << endl;

cout << "3. Year of birth " << endl;

cout << "4. Country " << endl;

cout << "5. Town " << endl;

cout << "#High school#\n6. Name " << endl;

int input;

while (1)

{

input = \_getch();

if (input == 27)

return arrayOfStruct;

input -= 48;

if (input < 1 || input > 6)

continue;

else break;

}

system("cls");

if (input == 1)

{

char lastName[1024];

cout << "Last name: ";

cin >> lastName;

searchResult(arrayOfStruct, sizeOfArrayOfStruct, lastName, functionForSearchString[0]);

}

if (input == 2)

{

int gender;

cout << "Gender(1 - male, 2 - female): ";

cin >> gender;

searchResult(arrayOfStruct, sizeOfArrayOfStruct, gender, functionForSearchInt[0]);

}

if (input == 3)

{

int yearOfBirth;

cout << "Year of birth: ";

cin >> yearOfBirth;

searchResult(arrayOfStruct, sizeOfArrayOfStruct, yearOfBirth, functionForSearchInt[1]);

}

if (input == 4)

{

char country[1024];

cout << "Country: ";

cin >> country;

searchResult(arrayOfStruct, sizeOfArrayOfStruct, country, functionForSearchString[1]);

}

if (input == 5)

{

char town[1024];

cout << "Town: ";

cin >> town;

searchResult(arrayOfStruct, sizeOfArrayOfStruct, town, functionForSearchString[2]);

}

if (input == 6)

{

char name[1024];

cout << "Name: ";

cin >> name;

searchResult(arrayOfStruct, sizeOfArrayOfStruct, name, functionForSearchString[3]);

}

}

}

void searchResult( student\* arrayOfStruct, int \*sizeOfArrayOfStruct,

char stringToCompare[1024], bool(\*funcIf)(student\*, int, char[1024]))

{

int now = -1; int i = 0;

bool right = true;

bool notFound = false;

while (1)

{

while (funcIf(arrayOfStruct, i, stringToCompare))

{

if (i == \*sizeOfArrayOfStruct)

{

if (now == -1)

{

system("cls");

cout << "Not found!" << endl;

notFound = true;

system("pause");

break;

}

i = now;

break;

}

if (i == -1)

{

i = now; break;

}

(right) ? (i++) : (i--);

}

if (notFound)

break;

system("cls");

outputStruct(arrayOfStruct[i]);

now = i;

int input2, br = 0;

while (1)

{

input2 = \_getch();

if (input2 == 27)

{

br = 1; break;

}

else if (input2 == 75 && i != 0)

{

i--;

right = false;

break;

}

else if (input2 == 77 && i != (\*sizeOfArrayOfStruct - 1))

{

i++;

right = true;

break;

}

}

if (br)

break;

}

}

void searchResult( student\* arrayOfStruct, int \*sizeOfArrayOfStruct,

int intToCompare, bool(\*funcIf)(student\*, int, int))

{

int now = -1; int i = 0;

bool right = true;

bool notFound = false;

while (1)

{

while (funcIf(arrayOfStruct, i, intToCompare))

{

if (i == \*sizeOfArrayOfStruct)

{

if (now == -1)

{

system("cls");

cout << "Not found!" << endl;

notFound = true;

system("pause");

break;

}

i = now;

break;

}

if (i == -1)

{

i = now; break;

}

(right) ? (i++) : (i--);

}

if (notFound)

break;

system("cls");

outputStruct(arrayOfStruct[i]);

now = i;

int input2, br = 0;

while (1)

{

input2 = \_getch();

if (input2 == 27)

{

br = 1; break;

}

else if (input2 == 75 && i != 0)

{

i--;

right = false;

break;

}

else if (input2 == 77 && i != (\*sizeOfArrayOfStruct - 1))

{

i++;

right = true;

break;

}

}

if (br)

break;

}

}

student\* sortStructArray(student\* arrayOfStruct, int \*sizeOfArrayOfStruct)

{

while (1)

{

system("cls");

cout << "1. Last name " << endl;

cout << "2. Town" << endl;

cout << "3. Year of birth" << endl;

int input; bool ret = false;

while (1)

{

input = \_getch();

if (input == 27)

{

ret = true;

break;

}

input -= 48;

if (input > 0 && input < 4)

break;

}

if (ret)

break;

sort(arrayOfStruct, sizeOfArrayOfStruct, input);

break;

}

system("cls");

cout << "Success!";

Sleep(1000);

system("cls");

return arrayOfStruct;

}

void sort(student\* arrayOfStruct, int \*sizeOfArrayOfStruct, int type)

{

if (type == 1)

{

int numOfSwaps;

do

{

numOfSwaps = 0;

for(int i = 1; i < \*sizeOfArrayOfStruct; i++)

if (arrayOfStruct[i].fullName.lastName[0] < arrayOfStruct[i - 1].fullName.lastName[0])

{

numOfSwaps++;

char var[1024];

strcpy\_s(var, arrayOfStruct[i].fullName.lastName);

strcpy\_s(arrayOfStruct[i].fullName.lastName, arrayOfStruct[i - 1].fullName.lastName);

strcpy\_s(arrayOfStruct[i - 1].fullName.lastName, var);

int n;

n = arrayOfStruct[i].numOfStud;

arrayOfStruct[i].numOfStud = arrayOfStruct[i - 1].numOfStud;

arrayOfStruct[i - 1].numOfStud = n;

}

} while (numOfSwaps != 0);

}

else if (type == 2)

{

int numOfSwaps;

do

{

numOfSwaps = 0;

for (int i = 1; i < \*sizeOfArrayOfStruct; i++)

if (arrayOfStruct[i].address.town[0] < arrayOfStruct[i - 1].address.town[0])

{

numOfSwaps++;

char var[1024];

strcpy\_s(var, arrayOfStruct[i].address.town);

strcpy\_s(arrayOfStruct[i].address.town, arrayOfStruct[i - 1].address.town);

strcpy\_s(arrayOfStruct[i - 1].address.town, var);

int n;

n = arrayOfStruct[i].numOfStud;

arrayOfStruct[i].numOfStud = arrayOfStruct[i - 1].numOfStud;

arrayOfStruct[i - 1].numOfStud = n;

}

} while (numOfSwaps != 0);

}

else if (type == 3)

{

int numOfSwaps;

do

{

numOfSwaps = 0;

for (int i = 1; i < \*sizeOfArrayOfStruct; i++)

if (arrayOfStruct[i].dateOfBirth.year < arrayOfStruct[i - 1].dateOfBirth.year)

{

numOfSwaps++;

int var;

var = arrayOfStruct[i].dateOfBirth.year;

arrayOfStruct[i].dateOfBirth.year = arrayOfStruct[i - 1].dateOfBirth.year;

arrayOfStruct[i - 1].dateOfBirth.year = var;

int n;

n = arrayOfStruct[i].numOfStud;

arrayOfStruct[i].numOfStud = arrayOfStruct[i - 1].numOfStud;

arrayOfStruct[i - 1].numOfStud = n;

}

} while (numOfSwaps != 0);

}

}

student\* finishProgram(student\* arrayOfStruct, int \*sizeOfArrayOfStruct)

{

system("cls");

cout << "Good luck!";

Sleep(1000);

exit(0);

}

student\* biggerArray(student\* arrayOfStruct, int\* sizeOfArrayOfStruct)

{

student \* arrayOfStructResult;

(\*sizeOfArrayOfStruct)++;

system("cls");

int sizeOfResultArray = \*sizeOfArrayOfStruct;

arrayOfStructResult = new student[sizeOfResultArray];

for (int i = 0; i < (sizeOfResultArray - 1); i++)

arrayOfStructResult[i] = arrayOfStruct[i];

return arrayOfStructResult;

}

bool gender(student\* arrayOfStruct, int numOfStruct, int intToCompare)

{

return (arrayOfStruct[numOfStruct].gender != intToCompare);

}

bool yearOfBirth(student\* arrayOfStruct, int numOfStruct, int intToCompare)

{

return (arrayOfStruct[numOfStruct].dateOfBirth.year != intToCompare);

}

bool lastName(student\* arrayOfStruct, int numOfStruct, char stringToCompare[1024])

{

return strcmp(arrayOfStruct[numOfStruct].fullName.lastName, stringToCompare);

}

bool country(student\* arrayOfStruct, int numOfStruct, char stringToCompare[1024])

{

return strcmp(arrayOfStruct[numOfStruct].address.country, stringToCompare);

}

bool town(student\* arrayOfStruct, int numOfStruct, char stringToCompare[1024])

{

return strcmp(arrayOfStruct[numOfStruct].address.town, stringToCompare);

}

bool highSchoolName(student\* arrayOfStruct, int numOfStruct, char stringToCompare[1024])

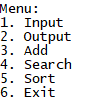
{

return strcmp(arrayOfStruct[numOfStruct].highSchool.name, stringToCompare);

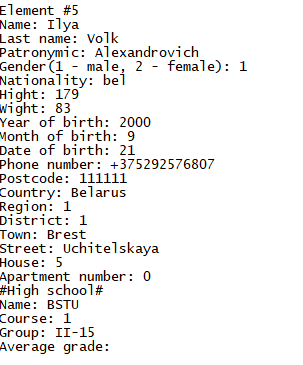
}

Результаты выполнения:

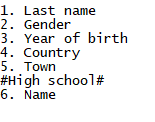
Menu)



Input)



Search)



Sort)



Вывод: научился работать с указателями на функции. Улучшил умения работы с динамическими массивами структур и функциями.

Блок схема функции **student\* biggerArray(student\* arrayOfStruct, int\* sizeOfArrayOfStruct)**

