Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра программной инженерии

 Лабораторная работа № 4

По дисциплине «Основы алгоритмизации и программирования»

На тему «Представление информации в виде структуры»

Выполнил:

Кулешов Артём

Студент 1 курса 8 группы

Преподаватель: асс. Андронова М.В.

Минск, 2024

4. В соответствии со своим вариантом разработать программу для данных, приведенных в таблице ниже. Определить структурированный тип, разработать меню для работы с ***массивом структур***.

В программу должны войти функции:

* ввод элементов структуры с клавиатуры;
* вывод элементов структуры в консольное окно;
* удаление заданной структурированной переменной;
* поиск информации;
* запись информации в файл;
* чтение данных из файла.

**Вариант 5(основной)**

**Ломбард.** База хранимых товаров и недвижимости: анкетные данные клиента, наименование товара, оценочная стоимость; сумма, выданная под залог, дата сдачи, срок хранения. Выбор по наименованию товара.

#include <iostream>

#include <string>

#include <fstream>

#include <Windows.h>

using namespace std;

void addClient();

void showClients();

void deleteClient();

void searchingClient();

void writeToFile();

void readFromFile();

int main()

{

int choice;

do {

cout << "Menu: " << endl;

cout << "1. Enter client data: " << endl;

cout << "2. Client data output: " << endl;

cout << "3. Removing a client: " << endl;

cout << "4. Searching the client: " << endl;

cout << "5. Enter information into file: " << endl;

cout << "6. Read information from file: " << endl;

cout << "7. Exit the program: " << endl;

cin >> choice;

switch (choice) {

case 1: addClient();

break;

case 2: showClients();

break;

case 3: deleteClient();

break;

case 4: searchingClient();

break;

case 5: writeToFile();

break;

case 6: readFromFile();

break;

case 7:

cout << "Exit the program: " << endl;

break;

default:

cout << "erRor" << endl;

break;

}

} while (choice != 7);

return 0;

}

//структура//

struct Client {

string clientsPersonalData;

string nameOfProduct;

string assessedValue;

string bailAmount;

string dueDate;

string shelfLife;

};

const int maxClients = 100;

Client clients[maxClients];

int clientCount = 1;

//функция для добавления клиента//

void addClient() {

if (clientCount > maxClients) {

cout << "Limit of clients." << endl;

exit(7);

}

Client newClient;

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

{

newClient.clientsPersonalData = "Enter the clients personal information: ";

cout << newClient.clientsPersonalData << endl;

cin >> clients[i].clientsPersonalData;

newClient.nameOfProduct = "Enter product name: ";

cout << newClient.nameOfProduct << endl;

cin >> clients[i].nameOfProduct;

newClient.assessedValue = "Enter estimated value: ";

cout << newClient.assessedValue << endl;

cin >> clients[i].assessedValue;

newClient.bailAmount = "Enter the bail amount: ";

cout << newClient.bailAmount << endl;

cin >> clients[i].bailAmount;

newClient.dueDate = "Enter due date: ";

cout << newClient.dueDate << endl;

cin >> clients[i].dueDate;

newClient.shelfLife = "Enter storage period: ";

cout << newClient.shelfLife << endl;

cin >> clients[i].shelfLife;

//clients[clientCount] = newClient;

//clientCount++;

}

cout << "Client added successfully." << endl;

}

//фукция для просмотра клиентов//

void showClients() {

/\*if (clientCount == 0) {

cout << "No clients." << endl;

exit(7);

}\*/

int clientCounter = 1;

for (int i = 0; i < clientCount; i++) {

cout << "Client" << clientCounter << ": " << endl;

cout << "Personal information of client: ";

cout << clients[i].clientsPersonalData << endl;

cout << "Product name: ";

cout << clients[i].nameOfProduct << endl;

cout << "Estimated value: ";

cout << clients[i].assessedValue << endl;

cout << "Bail amount: ";

cout << clients[i].bailAmount << endl;

cout << "Due date: ";

cout << clients[i].dueDate << endl;

cout << "Storage period: ";

cout << clients[i].shelfLife << endl;

}

}

//функция для удаления клиента//

void deleteClient() {

/\*if (clientCount == 0) {

cout << "No clients" << endl;

exit(7);

}\*/

int del;

cout << endl << "Enter number of client: ";

cin >> del;

for (int i = del - 1; i < clientCount - 1; i++)

{

clients[i] = clients[i + 1];

}

cout << endl << "Delete run success!" << endl;

clientCount -= 1;

}

//функция для поиска клиента по его персональным данным//

void searchingClient() {

if (clientCount == 0) {

cout << "No clients" << endl;

exit(7);

}

string clientsPersonalData;

cout << "Enter the clients personal information: " << endl;

cin >> clientsPersonalData;

bool found = false;

cout << "Results of the searching for '" << clientsPersonalData << "':" << endl;

for (int i = 0; i < clientCount; i++) {

if (clients[i].clientsPersonalData == clientsPersonalData) {

found = true;

cout << "Personal information of client: " << clients[i].clientsPersonalData << endl;

cout << "Estimated value: " << clients[i].assessedValue << endl;

cout << "Bail amount: " << clients[i].bailAmount << endl;

cout << "Due date: " << clients[i].dueDate << endl;

cout << "Storage period: " << clients[i].shelfLife << endl;

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

}

}

if (!found) {

cout << "Searching for '" << clientsPersonalData << "' failed." << endl;

}

}

//функция для записи в файл данных о клиенте//

void writeToFile()

{

ofstream file("file.txt");

if (file.is\_open()) {

for (int i = 0; i < clientCount; i++) {

file << clients[i].clientsPersonalData << "\n";

file << clients[i].nameOfProduct << "\n";

file << clients[i].assessedValue << "\n";

file << clients[i].bailAmount << "\n";

file << clients[i].dueDate << "\n";

file << clients[i].shelfLife << "\n";

}

cout << "Data upload success!" << endl;

file.close();

}

else {

cout << "Error of opening file." << endl;

}

}

//функция для чтения данных о клиенте из файла//

void readFromFile() {

ifstream file("file.txt");

if (file.is\_open()) {

clientCount = 0;

while (!file.eof()) {

Client newItem;

file >> newItem.clientsPersonalData;

file >> newItem.nameOfProduct;

file >> newItem.assessedValue;

file >> newItem.bailAmount;

file >> newItem.dueDate;

file >> newItem.shelfLife;

if (!file.fail()) {

clients[clientCount] = newItem;

clientCount++;

}

}

cout << "Data download success!" << endl;

file.close();

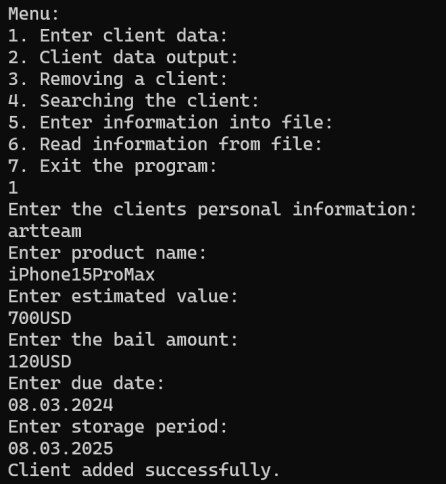
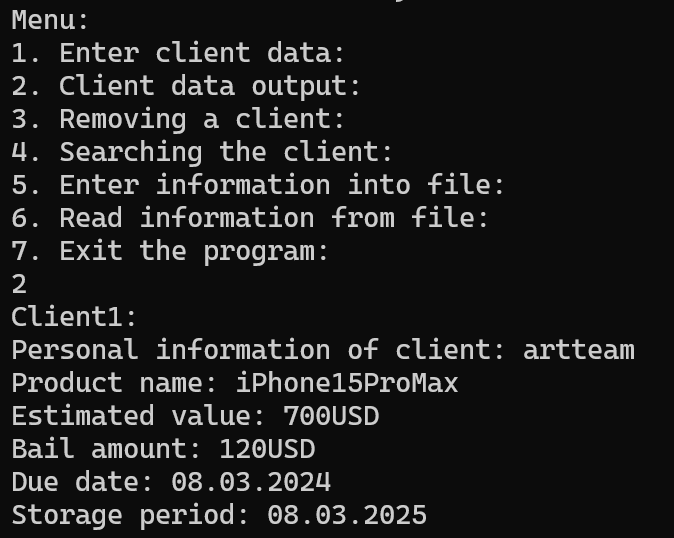
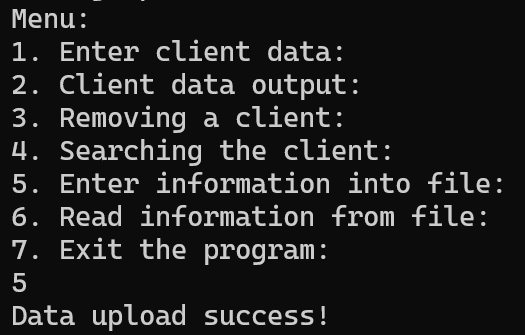
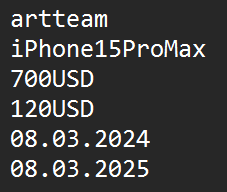
}

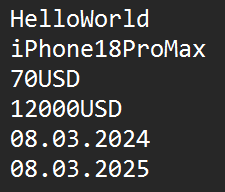
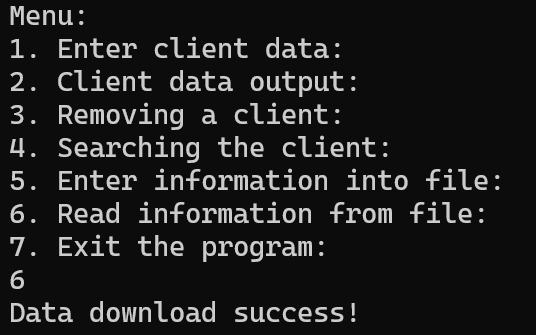
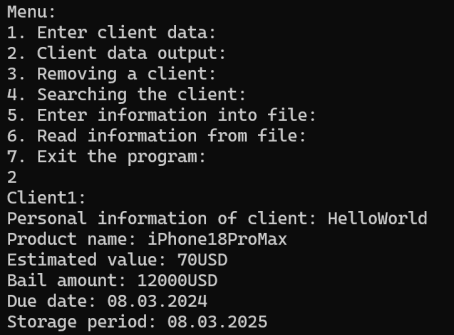
else {

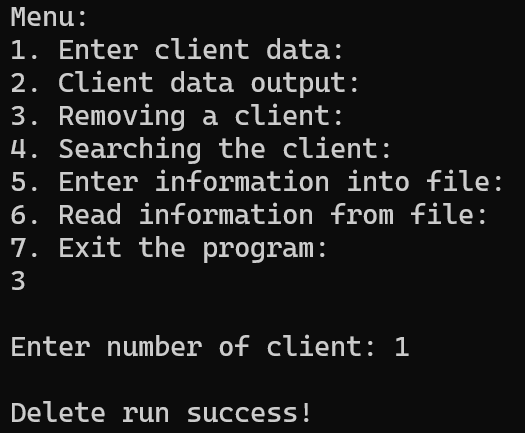
cout << "Error of opening file." << endl;

}

}



**Доп(1)**

1. Определить структуру для представления информации о сданных студентом экзаменах, содержащую поля: ФИО студента, число экзаменов, полученные оценки. Определить функции для обработки отдельного объекта (например, для проверки, сданы ли все экзамены на 4 и 5). Написать функцию для обработки массива структур. В результате обработки требуется вычислить характеристику успеваемости студентов, то есть отношение числа студентов, сдавших экзамены на 4 и 5, к общему числу студентов, в процентах.

#include <iostream>

#include <string>

#include <Windows.h>

#include <fstream>

#include <vector>

#define SIZE 10

#define OPTIONS 3

using namespace std;

//создаем структуру//

struct Exam {

string name;

string surname;

string patronymic;

int amountOfExem = 10;

vector<int> marks;

//создаем конструктор для нашей структуры

Exam(string name, string surname, string patronomyc, int amountOfExem, vector<int> marks)

{

this->name = name;

this->surname = surname;

this->patronymic = patronomyc;

this->amountOfExem = amountOfExem;

this->marks = marks;

}

};

//функция записи данных в структуру

void add(vector<Exam>& info)

{

//создаем необходимые переменные

string surname, name, patronomyc;

unsigned int amountOfExem;

vector<int> marks;

int temp;

cout << "Enter full name: "; cin >> surname >> name >> patronomyc;

cout << "Enter number of exams: "; cin >> amountOfExem;

cout << "Enter rating: ";

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

{

cin >> temp;

marks.push\_back(temp);

}

//записываем данные в структуру student

Exam student(name, surname, patronomyc, amountOfExem, marks);

info.push\_back(student);

}

//фунция вывода данных

void outInfo(vector<Exam>& info)

{

for (int i = 0; i < info.size(); i++)

{

cout << "Full name " << info[i].surname << " " << info[i].name << " " << info[i].patronymic << endl;

cout << "Number of exams: " << info[i].amountOfExem << endl << "Rating: ";

for (int j = 0; j < info[i].amountOfExem; j++)

{

cout << info[i].marks[j] << " ";

}

cout << endl;

}

}

//поиск элемента вектора по ФИО в структуре

int findInfo(vector<Exam>& info, char surname[], char name[], char patronymic[])

{

for (int i = 0; i < info.size(); i++)

{

if (info[i].name == name && info[i].surname == surname && info[i].patronymic == patronymic)

{

return i;

}

}

}

//функция для анализа данных

void dataFindProcessing(vector<Exam>& info, int number)

{

bool retake = false;

int count = 0;

for (int i = 0; i < info[number].amountOfExem; i++)

{

//при оценке ниже 4, ведем подсчет несданных экзаменов

if (info[number].marks[i] < 4)

{

count++;

retake = true;

}

}

if (retake)

{

cout << "Exams written below 4: " << count << endl;

}

else

{

cout << "All exams are written above 4!" << endl;

}

}

//функция анализа всех учеников на сдачу экзамена

void dataProcessing(vector<Exam>& info)

{

int count = 0;

bool retake = false;

for (int i = 0; i < info.size(); i++)

{

retake = false;

//перебираем оценки каждого студента и сравниваем

for (int j = 0; j < info[i].amountOfExem; j++)

{

if (info[i].marks[j] < 4)

{

retake = true;

}

}

if (!retake)

{

count++;

}

}

cout << (count / (double)info.size()) \* 100 << "% of students passed the exams." << endl;

}

void menu()

{

vector<Exam> students;

int answer = 0;

cout << "1 - Add. 2 - Conclusion. 3 - Find and check. 4 - All rating. 5 - Exit. " << endl;

while (answer != 5)

{

cout << "enter(choose): ";

cin >> answer;

switch (answer)

{

case 1:

add(students);

break;

case 2:

outInfo(students);

break;

case 3:

char name[33], surname[33], patronymic[33];

cout << "Enter the student's full name for verification: "; cin >> surname >> name >> patronymic;

dataFindProcessing(students, findInfo(students, surname, name, patronymic));

break;

case 4:

dataProcessing(students);

break;

}

}

}

int main()

{

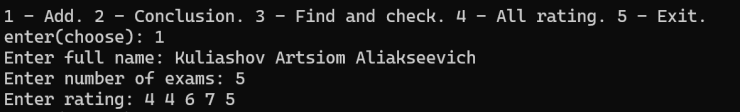
SetConsoleOutputCP(1251);

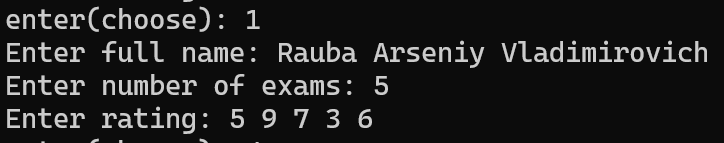
SetConsoleCP(1251);

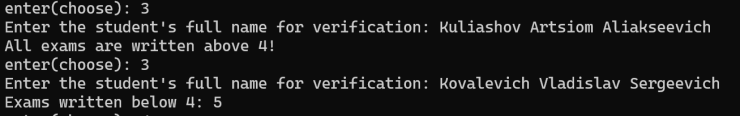
menu();

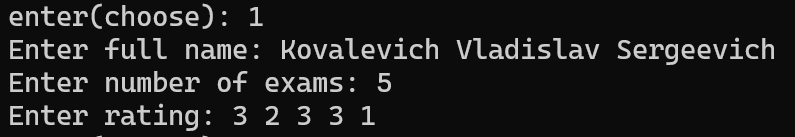
return 0;

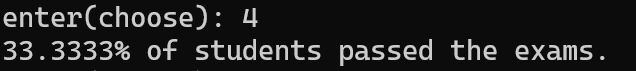
}

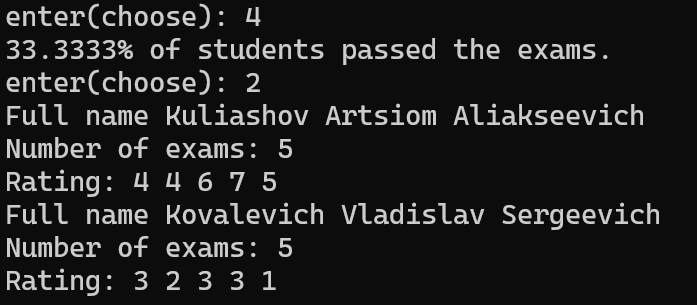












**Доп(2)**

2. Описать структуру с именем TRAIN, содержащую поля: названия пункта назначения, номер поезда, время отправления. Написать программу, выполняющую ввод с клавиатуры данных в массив, состоящий из восьми элементов типа TRAIN (записи должны быть размещены в алфавитном порядке по названиям пунктов назначения); вывод на экран информации о поездах, отправляющихся после введенного с клавиатуры времени (если таких поездов нет, то вывести сообщение об этом).

#include <iostream>

#include <Windows.h>

#include <fstream>

#include <string>

#include <vector>

#define SIZE 3

using namespace std;

//создаем структру//

struct Train

{

string name;

int number;

int hours;

int minutes;

//создаем конструктор для структуры//

Train(string name, int number, int hours, int minutes)

{

this->name = name;

this->number = number;

this->hours = hours;

this->minutes = minutes;

}

void swap(Train& second)

{

int itemp;

string stemp;

stemp = name;

name = second.name;

second.name = stemp;

itemp = number;

number = second.number;

second.number = itemp;

itemp = hours;

hours = second.hours;

second.hours = itemp;

itemp = minutes;

minutes = second.minutes;

second.minutes = itemp;

}

void print()

{

//вывод данных о станции//

cout << "Name of station : " << name << endl;

cout << "Train number : " << number << endl;

cout << "Train departure time: ";

if (hours < 10)

{

cout << "0";

}

cout << hours << ":";

if (minutes < 10)

{

cout << "0";

}

cout << minutes << endl << endl;

}

};

//функция для вывода информации из структуры//

void outInfo(vector<Train>& a)

{

for (int i = 0; i < a.size(); i++)

{

cout << "Station name: " << a[i].name << endl;

cout << "Train number: " << a[i].number << endl;

cout << "Train departure time: ";

if (a[i].hours < 10)

{

cout << "0";

}

cout << a[i].hours << ":";

if (a[i].minutes < 10)

{

cout << "0";

}

cout << a[i].minutes << endl << endl;

}

}

//функция добавления информации в структуру//

void add(vector<Train>& a)

{

bool out = false;

string name;

int number, hours, minutes;

for (int i = 0; out != true; i++)

{

cout << "Enter station name: "; cin >> name;

cout << "Enter train number: "; cin >> number;

cout << "Enter train departure time: "; cin >> hours >> minutes;

cout << endl;

if (hours > 24 || hours < 0)

{

cout << "Not matching time data, please re-enter." << endl;

i--;

continue;

}

if ((minutes < 0 || minutes > 60))

{

cout << "Not matching time data, please re-enter." << endl;

i--;

continue;

}

out = true;

}

//записываем данные в структуру train//

Train train(name, number, hours, minutes);

//добавляем данные в конец векторного массива структур//

a.push\_back(train);

}

//функция сортировки//

void sortLetter(vector<Train>& a)

{

int count = 0;

int itemp;

string stemp;

for (int i = 0; i < a.size() - 1; i++)

{

if (a[i].name[0] > a[i + 1].name[0])

{

a[i].swap(a[i + 1]);

}

}

for (int i = 0; i < a.size() - 1; i++)

{

if (a[i].name[0] > a[i + 1].name[0])

{

count++;

}

}

if (count != 0)

{

return sortLetter(a);

}

}

//поиск поездов, которые вызываются позже заданного времени//

void outTimeInfo(vector<Train>& a, int& hours, int& minutes)

{

int count = 0;

cout << "Trains that depart after the entered time: " << endl << endl;

for (int i = 0; i < a.size(); i++)

{

//при условии, если время из структуры больше заданного, то вызываем метод print//

if ((a[i].hours > hours))

{

a[i].print();

count++;

}

if ((a[i].hours == hours) && (a[i].minutes > minutes))

{

count++;

a[i].print();

}

}

if (count == 0)

{

cout << endl << "No trains found." << endl;

}

}

void menu()

{

vector<Train> info;

int answer = 0;

int hours = 0, minutes = 0;

cout << "1 - Add. 2 - Conclusion. 3 - Sort. 4 - Trains after time. 5 - Exit. " << endl;

while (answer != 5)

{

cout << "enter(choose): "; cin >> answer;

switch (answer)

{

case 1:

add(info);

break;

case 2:

outInfo(info);

break;

case 3:

sortLetter(info);

break;

case 4:

cout << "Enter time: "; cin >> hours >> minutes;

outTimeInfo(info, hours, minutes);

break;

}

}

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

menu();

return 0;

}

