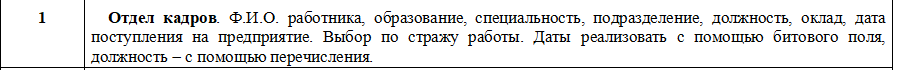
Банкузов Михаил 7 группа Лабораторная работа №5 вариант 1

Вариант 1





Задание 1

#include <iostream>

#include <vector>

#include <Windows.h>

#include <string>

#include <algorithm>

#include <ranges>

using namespace std;

// определяет структуру даты, представленной в виде битовых полей

struct day\_bytes

{

unsigned int week\_day : 3;

unsigned int day : 5;

unsigned int month : 4;

unsigned int year : 12;

};

// перечисления для должности

enum Dol

{

Программист = 1,

Инжинер,

Технолог

};

// описание структуры

struct citizen

{

std::string name\_surname;

std::string edu;

std::string spez;

std::string podrazd;

std::string oklad;

day\_bytes date;

Dol dol;

};

// определяет перегрузку оператора для вывода записи о работнике в стандартный поток вывода

std::ostream& operator<<(std::ostream& os, const citizen& citizens)

{

os << citizens.name\_surname << '\n' << citizens.edu << '\n' << citizens.spez << '\n' << citizens.podrazd << '\n' << citizens.oklad << '\n' << citizens.date.day << '.' << citizens.date.month << '.' << citizens.date.year << '\n';

switch (citizens.dol)

{

case 1: os << "Программист";

break;

case 2: os << "Инжинер";

break;

case 3: os << "Технолог";

break;

}

return os;

}

//функция для ввода данных о работниках в базу данных

void include\_citizens(std::vector<citizen>& guesses, int& n)

{

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

{

int count = NULL;

int dol = NULL;

int day = NULL, month = NULL, year = NULL;

cout << endl << "ФИО: "; cin >> guesses[i].name\_surname;

cout << endl << "Образование: "; cin >> guesses[i].edu;

cout << endl << "Специальность: "; cin >> guesses[i].spez;

cout << endl << "Подразделение: "; cin >> guesses[i].podrazd;

cout << endl << "Оклад: "; cin >> guesses[i].oklad;

cout << endl << "Дата:"; cin >> day; cin >> month; cin >> year;

guesses[i].date.day = day;

guesses[i].date.month = month;

guesses[i].date.year = year;

cout << endl << "Должность: Программист - 1, Инжинер - 2, Технолог - 3 "; cin >> dol;

guesses[i].dol = (Dol)dol;

}

}

//функция для поиска работников по дате

std::vector<citizen> search\_by\_date(const std::vector<citizen>& citizens, const day\_bytes& date)

{

std::vector<citizen> result;

for (const auto& c : citizens)

{

if (c.date.day == date.day && c.date.month == date.month && c.date.year == date.year)//если совпадает то выводит структуру

{

result.push\_back(c);

}

}

return result;

}

// выводит все структуры

void print\_all\_citizens(const std::vector<citizen>& citizens)

{

for (const auto& c : citizens) {

std::cout << c << std::endl;

}

}

// удаляет структуры по критерию (фамилии)

void delete\_citizen\_by\_surname(std::vector<citizen>& citizens)

{

std::string surname\_to\_delete;

cout << "Введите фамилию работника, которого хотите удалить: ";

cin >> surname\_to\_delete;

citizens.erase(std::remove\_if(citizens.begin(), citizens.end(), [surname\_to\_delete](const citizen& c) {return c.name\_surname == surname\_to\_delete; }), citizens.end());

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

std::vector<citizen> citizens;

int n = 0;

day\_bytes date;

int choice = 0;

do {

cout << "\nМеню:" << endl;

cout << "1. Ввод данных о работниках" << endl;

cout << "2. Поиск работников по дате" << endl;

cout << "3. Вывод всех работников" << endl;

cout << "4. Удаление работника по фамилии" << endl;

cout << "5. Выход" << endl;

cout << "Выберите действие: ";

cin >> choice;

switch (choice)

{

case 1:

cout << "Введите количество работников: ";

cin >> n;

citizens.resize(n);

include\_citizens(citizens, n);

break;

case 2:

std::vector<citizen> result = search\_by\_date(citizens, date);

cin >> date.day >> date.month >> date.year;

if (result.empty())

{

cout << "Работники не найдены." << endl;

}

else

{

print\_all\_citizens(result);

}

break;

case 3:

print\_all\_citizens(citizens);

break;

case 4:

delete\_citizen\_by\_surname(citizens);

break;

case 5:

cout << "До свидания!" << endl;

break;

default:

cout << "Некорректный ввод. Попробуйте еще раз." << endl;

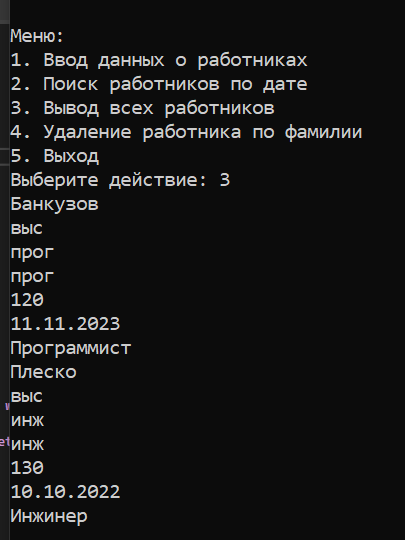
break;

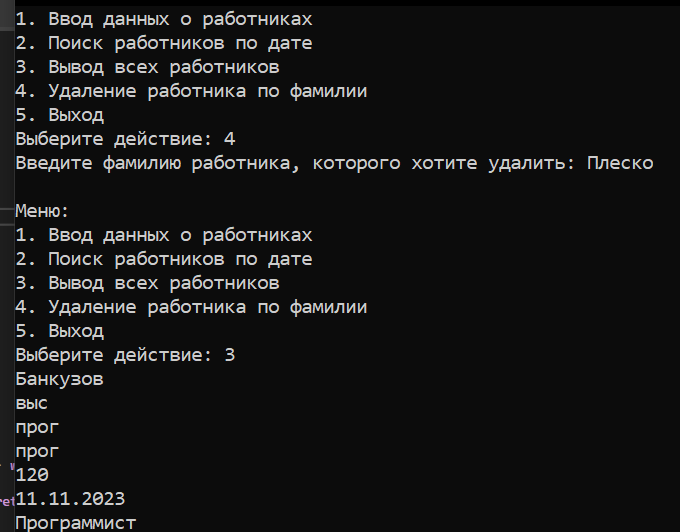
}

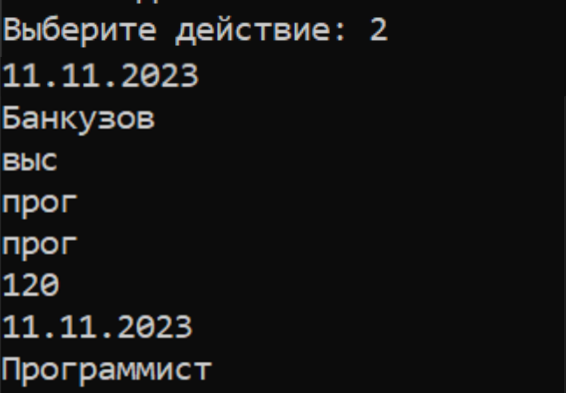
} while (choice != 5);

return 0;

}







Задание 2

#include <iostream>

#include <Windows.h>

#include <vector>

#include <algorithm>

#include <ranges>

using std::cout; using std::cin; using std::endl;

// Определение структуры данных для хранения информации об экзамене

struct ekzs

{

std::string name; // Фамилия

std::string ekz; // Название экзамена

std::string date; // Дата экзамена

friend std::ostream& operator<<(std::ostream& os, const ekzs& dt); // Перегрузка оператора вывода в поток

};

// Определение объединения для хранения структуры данных

union all\_ekzs

{

struct ekzs ekz;

};

// Перегрузка оператора вывода в поток для структуры данных

std::ostream& operator<<(std::ostream& os, const ekzs& dt)

{

os << dt.name << '\n' << dt.ekz << '\n' << dt.date << endl; // Вывод информации об экзамене

return os;

}

// Функция для ввода информации о экзаменах

void include\_ekz(std::vector<ekzs>& guesses, int& n)

{

for (int i = 0; i < n; i++) // Цикл для ввода информации о каждом экзамене

{

int count = NULL; // Обнуление переменной

cout << endl << "Фамилия: "; cin >> guesses[i].name; // Ввод фамилии студента

cout << endl << "Название экзамена: "; cin >> guesses[i].ekz; // Ввод названия экзамена

cout << endl << "Дата:"; cin >> guesses[i].date; // Ввод даты экзамена

}

}

// Функция для проверки информации об экзаменах по фамилии студента

void check\_ekz(std::vector<ekzs>& vector\_of\_trains, int& n)

{

std::string name; // Переменная для хранения фамилии студента

cout << "Введите фимилию: "; cin >> name; // Ввод фамилии студента для поиска информации об экзаменах

for (int i = 0; i < n; i++) // Цикл для поиска информации об экзаменах по фамилии студента

{

if (vector\_of\_trains[i].name == name) // Если фамилия совпадает

{

cout << vector\_of\_trains[i]; // Вывод информации об экзамене

}

}

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

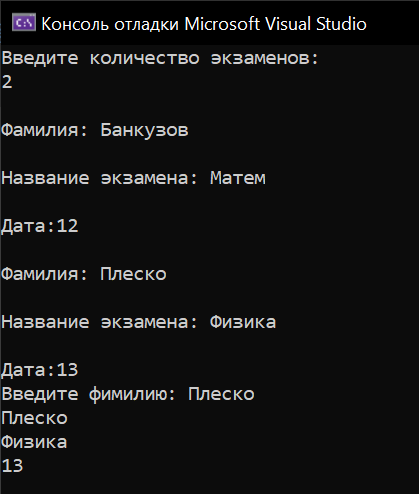
int n = NULL; cout << "Введите количество экзаменов: " << endl; cin >> n;

std::vector<ekzs> vector\_of\_ekzs(n);

include\_ekz(vector\_of\_ekzs, n);

check\_ekz(vector\_of\_ekzs, n);

}



Дополнительные задания

Вариант 2





Задание 1

#include <iostream>

#include <vector>

#include <Windows.h>

#include <string>

#include <algorithm>

#include <ranges>

using namespace std;

// определяет структуру даты, представленной в виде битовых полей

struct day\_bytes{

unsigned int week\_day : 3;

unsigned int day : 5;

unsigned int month : 4;

unsigned int year : 12;

};

// перечисления для гендера

enum Gender{

Мужской = 1,

Женский,

Вертолет

};

// описание структуры

struct citizen{

std::string name\_surname;

std::string adres;

day\_bytes date;

Gender gender;

};

// определяет перегрузку оператора для вывода записи о работнике в стандартный поток вывода

std::ostream& operator<<(std::ostream& os, const citizen& citizens){

os << citizens.name\_surname << '\n' << citizens.adres << '\n' << citizens.date.day << '.' << citizens.date.month << '.' << citizens.date.year << '\n';

switch (citizens.gender)

{

case 1: os << "Мужской";

break;

case 2: os << "Женский";

break;

case 3: os << "Вертолет";

break;

}

return os;

}

//функция для ввода данных о работниках в базу данных

void include\_citizens(std::vector<citizen>& guesses, int& n){

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

{

int count = NULL;

int gender = NULL;

int day = NULL, month = NULL, year = NULL;

cout << endl << "ФИО: "; cin >> guesses[i].name\_surname;

cout << endl << "Адрес проживания: "; cin >> guesses[i].adres;

cout << endl << "Дата рождения:"; cin >> day; cin >> month; cin >> year;

guesses[i].date.day = day;

guesses[i].date.month = month;

guesses[i].date.year = year;

cout << endl << "Гендер: мужской - 1, женский - 2, боевой вертолёт - 3 "; cin >> gender;

guesses[i].gender = (Gender)gender;

}

}

//функция для поиска работников по дате

std::vector<citizen> search\_by\_date(const std::vector<citizen>& citizens, const day\_bytes& date){

std::vector<citizen> result;

for (const auto& c : citizens)

{

if (c.date.day == date.day && c.date.month == date.month && c.date.year == date.year)//если совпадает то выводит структуру

{

result.push\_back(c);

}

}

return result;

}

// выводит все структуры

void print\_all\_citizens(const std::vector<citizen>& citizens){

for (const auto& c : citizens) {

std::cout << c << std::endl;

}

}

// удаляет структуры по критерию (фамилии)

void delete\_citizen\_by\_surname(std::vector<citizen>& citizens){

std::string surname\_to\_delete;

cout << "Введите фамилию работника, которого хотите удалить: ";

cin >> surname\_to\_delete;

citizens.erase(std::remove\_if(citizens.begin(), citizens.end(), [surname\_to\_delete](const citizen& c) {return c.name\_surname == surname\_to\_delete; }), citizens.end());

}

int main(){

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

std::vector<citizen> citizens;

int n = 0;

day\_bytes date;

int choice = 0;

do {

cout << "\nМеню:" << endl;

cout << "1. Ввод данных о гражданине" << endl;

cout << "2. Поиск граждан по дате" << endl;

cout << "3. Вывод всех граждан" << endl;

cout << "4. Удаление гражданина по фамилии" << endl;

cout << "5. Выход" << endl;

cout << "Выберите действие: ";

cin >> choice;

switch (choice)

{

case 1:

cout << "Введите количество работников: ";

cin >> n;

citizens.resize(n);

include\_citizens(citizens, n);

break;

case 2:

std::vector<citizen> result = search\_by\_date(citizens, date);

cin >> date.day >> date.month >> date.year;

if (result.empty())

{

cout << "Работники не найдены." << endl;

}

else

{

print\_all\_citizens(result);

}

break;

case 3:

print\_all\_citizens(citizens);

break;

case 4:

delete\_citizen\_by\_surname(citizens);

break;

case 5:

cout << "До свидания!" << endl;

break;

default:

cout << "Некорректный ввод. Попробуйте еще раз." << endl;

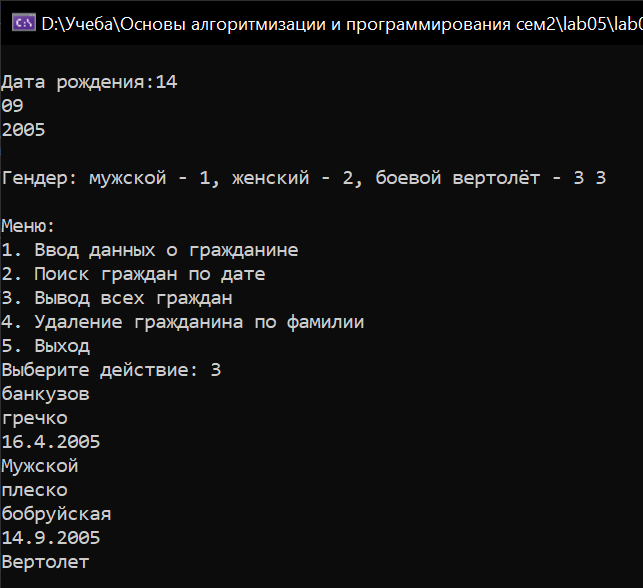
break;

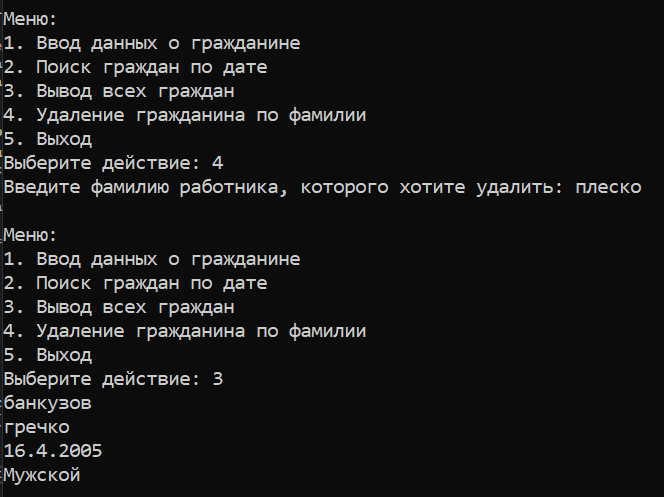
}

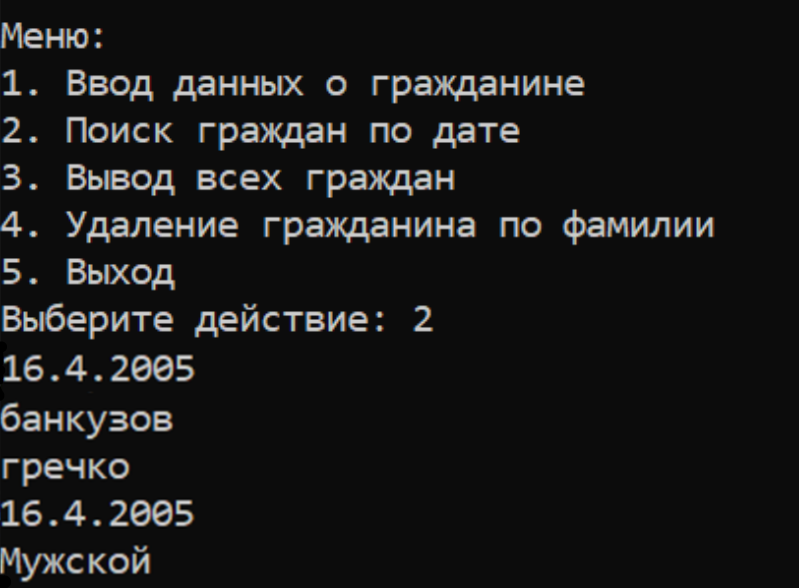
} while (choice != 5);

return 0;

}







Задание 2

#include <iostream>

#include <Windows.h>

#include <vector>

#include <algorithm>

#include <ranges>

using std::cout; using std::cin; using std::endl;

// Определение структуры данных для хранения информации об поездах

struct trains

{

std::string name;

std::string number;

std::string time;

friend std::ostream& operator<<(std::ostream& os, const trains& dt);//Перегрузка оператора вывода в поток

};

// Определение объединения для хранения структуры данных

union all\_trains

{

struct trains train;

};

// Перегрузка оператора вывода в поток для структуры данных

std::ostream& operator<<(std::ostream& os, const trains& dt)

{

os << dt.name << '\n' << dt.number << '\n' << dt.time << endl;

return os;

}

// Функция для ввода информации о экзаменах

void include\_trains(std::vector<trains>& guesses, int& n)

{

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

{

int count = NULL;

cout << endl << "Название пункта назначения: "; cin >> guesses[i].name;

cout << endl << "Номер поезда: "; cin >> guesses[i].number;

cout << endl << "Время отправления:"; cin >> guesses[i].time;

}

}

// Функция для проверки информации

void check\_trains(std::vector<trains>& vector\_of\_trains, int& n)

{

std::string name;

cout << "Введите пункт назначения: "; cin >> name;

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

{

if (vector\_of\_trains[i].name == name)

{

cout << vector\_of\_trains[i];

}

}

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

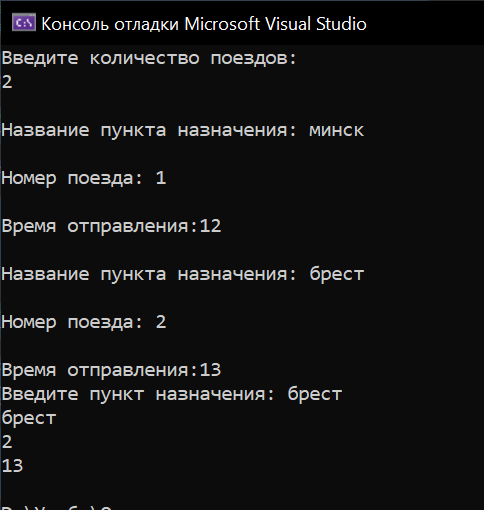
int n = NULL; cout << "Введите количество поездов: " << endl; cin >> n;

std::vector<trains> vector\_of\_trains(n);

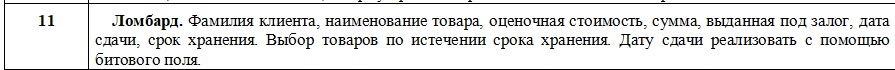
include\_trains(vector\_of\_trains, n);

check\_trains(vector\_of\_trains, n);

}



Вариант 11





Задание 1

#include <iostream>

#include <vector>

#include <Windows.h>

#include <string>

#include <algorithm>

#include <ranges>

using namespace std;

// определяет структуру даты, представленной в виде битовых полей

struct day\_bytes{

unsigned int week\_day : 3;

unsigned int day : 5;

unsigned int month : 4;

unsigned int year : 12;

};

// описание структуры

struct citizen{

std::string name\_surname;

std::string product;

int mark\_size;

int sum;

day\_bytes date\_of\_return;

int shelf\_life;

};

// определяет перегрузку оператора для вывода записи о работнике в стандартный поток вывода

std::ostream& operator<<(std::ostream& os, const citizen& citizens){

os << citizens.name\_surname << '\n' << citizens.product << '\n' << citizens.mark\_size << '\n' << citizens.sum

<< '\n' << citizens.shelf\_life << '\n' << citizens.date\_of\_return.day << '.' << citizens.date\_of\_return.month

<< '.' << citizens.date\_of\_return.year << '\n';

return os;

}

//функция для ввода данных о работниках в базу данных

void include\_citizens(std::vector<citizen>& guesses, int& n){

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

{

int count = NULL;

int gender = NULL;

int day = NULL, month = NULL, year = NULL;

cout << endl << "ФИО: "; cin >> guesses[i].name\_surname;

cout << endl << "Наименование товара: "; cin >> guesses[i].product;

cout << endl << "Оценочная стоимость: "; cin >> guesses[i].mark\_size;

cout << endl << "Сумма, выданная под залог: "; cin >> guesses[i].sum;

cout << endl << "Дата сдачи: "; cin >> day; cin >> month; cin >> year;

guesses[i].date\_of\_return.day = day;

guesses[i].date\_of\_return.month = month;

guesses[i].date\_of\_return.year = year;

cout << endl << "Срок хранения: "; cin >> guesses[i].shelf\_life;

}

}

//функция для поиска товара по дате

std::vector<citizen> search\_by\_date(const std::vector<citizen>& citizens, const day\_bytes& date) {

std::vector<citizen> result;

for (const auto& c : citizens)

{

if (c.date\_of\_return.day == date.day && c.date\_of\_return.month == date.month && c.date\_of\_return.year == date.year)//если совпадает то выводит структуру

{

result.push\_back(c);

}

}

return result;

}

// выводит все структуры

void print\_all\_citizens(const std::vector<citizen>& citizens){

for (const auto& c : citizens) {

std::cout << c << std::endl;

}

}

// удаляет структуры по критерию (фамилии)

void delete\_citizen\_by\_surname(std::vector<citizen>& citizens){

std::string surname\_to\_delete;

cout << "Введите фамилию товара, которого хотите удалить: ";

cin >> surname\_to\_delete;

citizens.erase(std::remove\_if(citizens.begin(), citizens.end(), [surname\_to\_delete](const citizen& c) {return c.name\_surname == surname\_to\_delete; }), citizens.end());

}

int main(){

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

std::vector<citizen> citizens;

int n = 0;

day\_bytes date;

int choice = 0;

do {

cout << "\nМеню:" << endl;

cout << "1. Ввод данных о товарах" << endl;

cout << "2. Поиск товара по дате" << endl;

cout << "3. Вывод всех товаров" << endl;

cout << "4. Удаление товара по фамилии" << endl;

cout << "5. Выход" << endl;

cout << "Выберите действие: ";

cin >> choice;

switch (choice)

{

case 1:

cout << "Введите количество товаров: ";

cin >> n;

citizens.resize(n);

include\_citizens(citizens, n);

break;

case 2:

std::vector<citizen> result = search\_by\_date(citizens, date);

cin >> date.day >> date.month >> date.year;

if (result.empty())

{

cout << "товары не найдены." << endl;

}

else

{

print\_all\_citizens(result);

}

break;

case 3:

print\_all\_citizens(citizens);

break;

case 4:

delete\_citizen\_by\_surname(citizens);

break;

case 5:

cout << "До свидания!" << endl;

break;

default:

cout << "Некорректный ввод. Попробуйте еще раз." << endl;

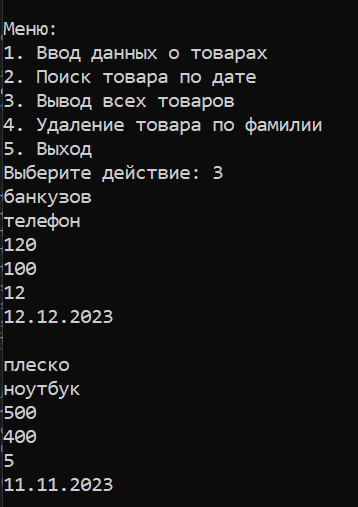
break;

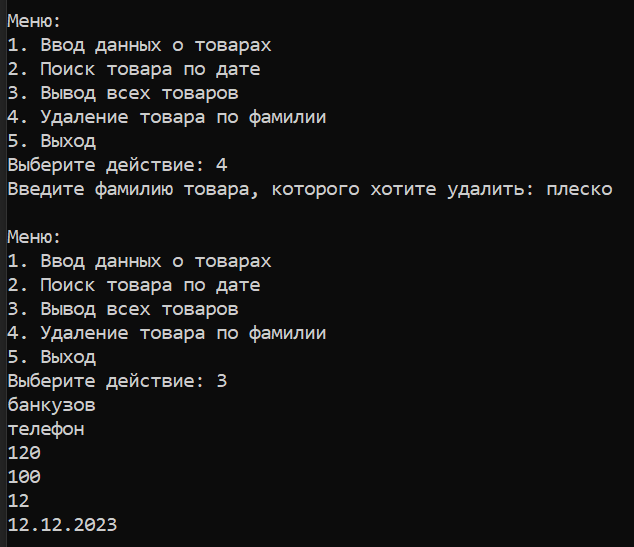
}

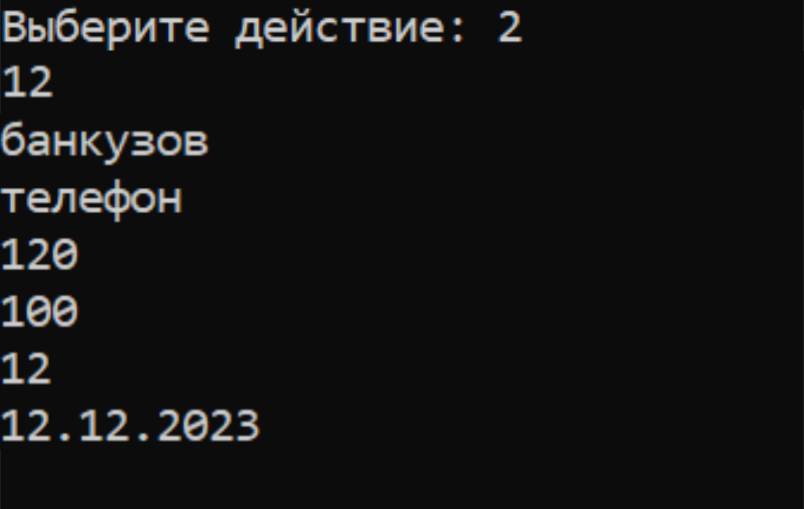
} while (choice != 5);

return 0;

}







Задание 2

#include <iostream>

#include <Windows.h>

#include <vector>

#include <algorithm>

#include <ranges>

using std::cout; using std::cin; using std::endl;

// определяет структуру даты, представленной в виде битовых полей

struct date

{

unsigned day : 5;

unsigned month : 4;

unsigned year : 12;

};

// Определение структуры данных для хранения информации об авто

struct trains

{

std::string name;

std::string number;

std::string color;

date time;

std::string type;

date last\_time;

std::string own;

friend std::ostream& operator<<(std::ostream& os, const trains& dt);

};

// Определение объединения для хранения структуры данных

union all\_trains

{

struct trains train;

};

// Перегрузка оператора вывода в поток для структуры данных

std::ostream& operator<<(std::ostream& os, const trains& dt)

{

os << dt.name << '\n' << dt.number << '\n' << dt.color << '\n' << dt.time.day << '.' << dt.time.month << '.' << dt.time.year << '\n' << dt.type << '\n' <<

dt.last\_time.day << '.' << dt.last\_time.month << '.' << dt.last\_time.year << endl << dt.own << endl;

return os;

}

// Функция для ввода информации о авто

void include\_trains(std::vector<trains>& guesses, int& n)

{

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

{

int count = NULL;

int day = NULL, month = NULL, year = NULL;

cout << endl << "марка автомобиля: "; cin >> guesses[i].name;

cout << endl << "Цвет машины: "; cin >> guesses[i].color;

cout << endl << "Заводской номер: "; cin >> guesses[i].number;

cout << endl << "дата выпуска:"; cin >> day >> month >> year;

guesses[i].time.day = day;

guesses[i].time.month = month;

guesses[i].time.year = year;

cout << endl << "Тип кузова: "; cin >> guesses[i].type;

cout << endl << "Пследний тех. осмотр: "; cin >> day >> month >> year;

guesses[i].last\_time.day = day;

guesses[i].last\_time.month = month;

guesses[i].last\_time.year = year;

cout << endl << "Владелец: "; cin >> guesses[i].own;

}

}

// Функция для проверки информации по владельцу

void check\_trains(std::vector<trains>& vector\_of\_trains, int& n)

{

std::string own;

cout << "Введите ФИО владельца: "; cin >> own;

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

{

if (vector\_of\_trains[i].own == own)

{

cout << vector\_of\_trains[i];

}

}

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

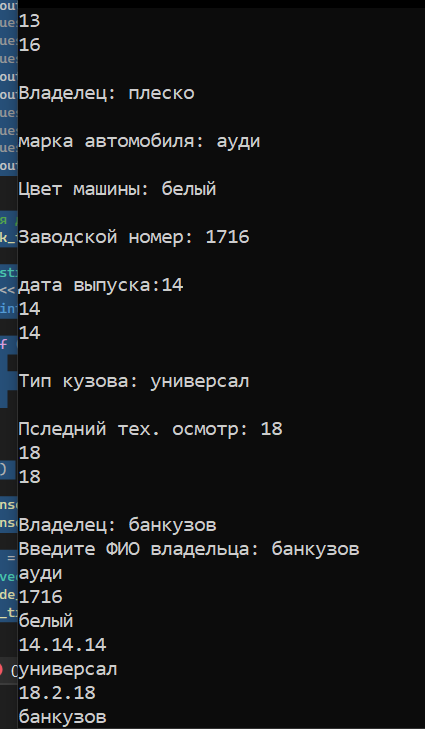
int n = NULL; cout << "Введите количество машин: " << endl; cin >> n;

std::vector<trains> vector\_of\_trains(n);

include\_trains(vector\_of\_trains, n);

check\_trains(vector\_of\_trains, n);

}



Вариант 13





Задание 1

#include <iostream>

#include <vector>

#include <Windows.h>

#include <string>

#include <algorithm>

#include <ranges>

using namespace std;

// определяет структуру даты, представленной в виде битовых полей

struct day\_bytes{

unsigned int week\_day : 3;

unsigned int day : 5;

unsigned int month : 4;

unsigned int year : 12;

};

// перечисления для гендера

enum Gender{

фит = 1,

тов,

хтит

};

// описание структуры

struct citizen{

std::string name\_surname;

day\_bytes date;

std::string speciality;

Gender faculty;

double sr\_mark;

};

// определяет перегрузку оператора для вывода записи о работнике в стандартный поток вывода

std::ostream& operator<<(std::ostream& os, const citizen& citizens){

os << citizens.name\_surname << '\n' << citizens.speciality << '\n' << citizens.date.day << '.' << citizens.date.month << '.' << citizens.date.year << '\n' << citizens.sr\_mark << "\n";

switch (citizens.faculty)

{

case 1: os << "фит";

break;

case 2: os << "тов";

break;

case 3: os << "хтит";

break;

}

return os;

}

//функция для ввода данных о работниках в базу данных

void include\_citizens(std::vector<citizen>& guesses, int& n){

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

{

int count = NULL;

int gender = NULL;

int day = NULL, month = NULL, year = NULL;

cout << endl << "ФИО: "; cin >> guesses[i].name\_surname;

cout << endl << "Специальность: "; cin >> guesses[i].speciality;

cout << endl << "Дата поступления:"; cin >> day; cin >> month; cin >> year;

guesses[i].date.day = day;

guesses[i].date.month = month;

guesses[i].date.year = year;

cout << endl << "Гендер: фит - 1, тов - 2, хтит - 3 "; cin >> gender;

guesses[i].faculty = (Gender)gender;

cout << endl << "Средний балл:"; cin >> guesses[i].sr\_mark;

guesses[i].faculty = (Gender)gender;

}

}

//функция для поиска студента по дате

std::vector<citizen> search\_by\_date(const std::vector<citizen>& citizens, const double& sr\_mark){

std::vector<citizen> result;

for (const auto& c : citizens)

{

if (c.sr\_mark == sr\_mark)//если совпадает то выводит структуру

{

result.push\_back(c);

}

}

return result;

}

// выводит все структуры

void print\_all\_citizens(const std::vector<citizen>& citizens){

for (const auto& c : citizens) {

std::cout << c << std::endl;

}

}

// удаляет структуры по критерию (фамилии)

void delete\_citizen\_by\_surname(std::vector<citizen>& citizens){

std::string surname\_to\_delete;

cout << "Введите фамилию студента, которого хотите удалить: ";

cin >> surname\_to\_delete;

citizens.erase(std::remove\_if(citizens.begin(), citizens.end(), [surname\_to\_delete](const citizen& c) {return c.name\_surname == surname\_to\_delete; }), citizens.end());

}

int main(){

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

std::vector<citizen> citizens;

int n = 0;

day\_bytes date;

int choice = 0;

do {

cout << "\nМеню:" << endl;

cout << "1. Ввод данных о студентах" << endl;

cout << "2. Поиск студента по дате" << endl;

cout << "3. Вывод всех студентов" << endl;

cout << "4. Удаление студента по фамилии" << endl;

cout << "5. Выход" << endl;

cout << "Выберите действие: ";

cin >> choice;

switch (choice)

{

case 1:

cout << "Введите количество студентов: ";

cin >> n;

citizens.resize(n);

include\_citizens(citizens, n);

break;

case 2:

std::vector<citizen> result = search\_by\_date(citizens, sr\_mark);

cin >> sr\_mark;

if (result.empty())

{

cout << "студент не найдены." << endl;

}

else

{

print\_all\_citizens(result);

}

break;

case 3:

print\_all\_citizens(citizens);

break;

case 4:

delete\_citizen\_by\_surname(citizens);

break;

case 5:

cout << "До свидания!" << endl;

break;

default:

cout << "Некорректный ввод. Попробуйте еще раз." << endl;

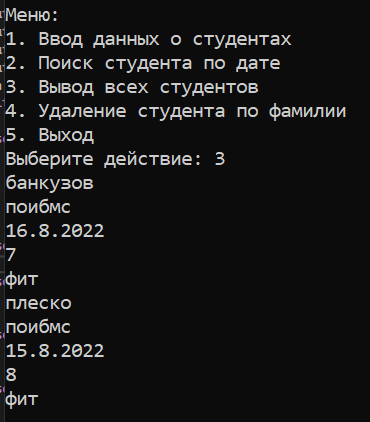
break;

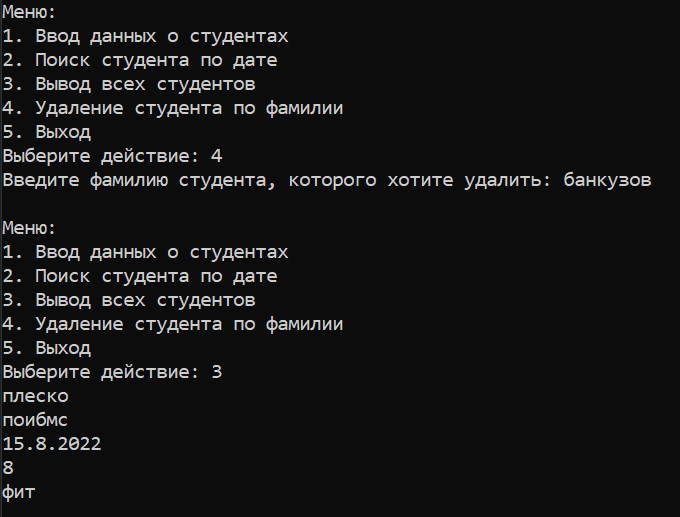
}

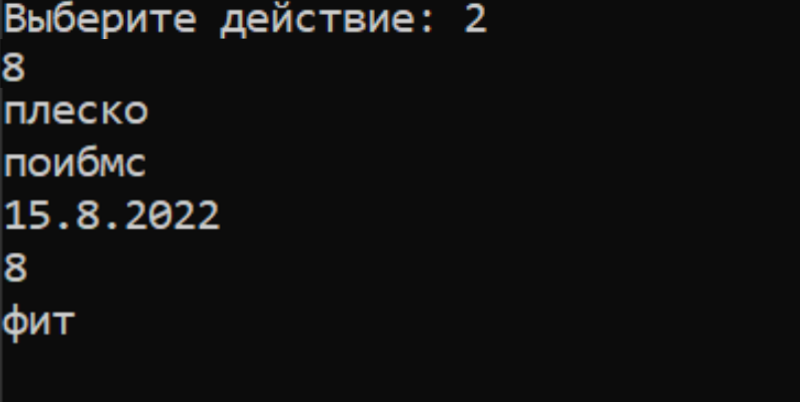
} while (choice != 5);

return 0;

}







Задание 2

#include <iostream>

#include <Windows.h>

#include <vector>

#include <algorithm>

#include <ranges>

using std::cout; using std::cin; using std::endl;

// Определение структуры данных для хранения информации о стране

struct trains

{

std::string name;

std::string capital;

int population;

long int size;

std::string president\_surname;

friend std::ostream& operator<<(std::ostream& os, const trains& dt);

};

// Перегрузка оператора вывода в поток для структуры данных

std::ostream& operator<<(std::ostream& os, const trains& dt)

{

os << dt.name << '\n' << dt.capital << '\n' << dt.population << '\n' << dt.size << '\n' << dt.president\_surname << endl;

return os;

}

// Функция для ввода информации о авто

void include\_trains(std::vector<trains>& guesses, int& n)

{

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

{

int count = NULL;

cout << endl << "Название страны: "; cin >> guesses[i].name;

cout << endl << "Название столицы: "; cin >> guesses[i].capital;

cout << endl << "Население:"; cin >> guesses[i].population;

cout << endl << "Площадь страны: "; cin >> guesses[i].size;

cout << endl << "Фамилия президента: "; cin >> guesses[i].president\_surname;

}

}

// Функция для проверки информации по названию

void check\_trains(std::vector<trains>& vector\_of\_trains, int& n)

{

std::string own;

cout << "Введите название: "; cin >> own;

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

{

if (vector\_of\_trains[i].name == own)

{

cout << vector\_of\_trains[i];

}

}

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int n = NULL; cout << "Введите количество стран: " << endl; cin >> n;

std::vector<trains> vector\_of\_trains(n);

include\_trains(vector\_of\_trains, n);

check\_trains(vector\_of\_trains, n);

}

