Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра программной инженерии

 Лабораторная работа № 5

По дисциплине «Основы алгоритмизации и программирования»

На тему «Объединения, перечисления, битовые поля»

Выполнил:

Студент 1 курса 10 группы

Жамойдо Артём Игоревич

Преподаватель: асс. Андронова М.В.

Минск, 2024

В соответствии со своим вариантом разработать программу с использованием ***перечислений*** и ***битовых*** *полей* для работы с данными из таблицы, приведенной ниже. Реализовать функции ввода с клавиатуры, вывода на экран, удаления, поиска элементов. Интерфейс пользователя осуществить в виде меню.

**Вариант 4**

**Клиенты банка**. Ф.И.О., тип счета (срочный, льготный и т. д.), номер счета, сумма на счете, дата последнего изменения. Выбор по диапазону суммы (<100, >100). Дату реализовать с помощью битового поля, тип счета − с помощью перечисления.

#include <iostream>

#include <string>

#include <vector>

#include <Windows.h>

using namespace std;

// Перечисление для типа счета

enum AccountType

{

SAVINGS,

CURRENT,

FIXED\_DEPOSIT,

OTHER

};

// Структура для хранения информации о клиенте

struct BankClient

{

string fullName;

AccountType accountType;

long accountNumber;

double balance;

// Битовое поле для хранения даты последнего изменения

struct Date

{

unsigned int day : 5;

unsigned int month : 4;

unsigned int year : 16;

}

lastModified;

};

// Функция для вывода информации о клиенте

void displayClient(const BankClient& client)

{

cout << "ФИО: " << client.fullName << endl;

cout << "Тип счета: ";

// Проверка типа счета

switch (client.accountType)

{

case SAVINGS:

cout << "Сберегательный" << endl;

break;

case CURRENT:

cout << "Текущий" << endl;

break;

case FIXED\_DEPOSIT:

cout << "Депозитный" << endl;

break;

case OTHER:

cout << "Другой" << endl;

break;

}

cout << "Номер счета: " << client.accountNumber << endl;

cout << "Сумма на счете: " << client.balance << endl;

cout << "Дата последнего изменения: " << client.lastModified.day << "." << client.lastModified.month << "." << client.lastModified.year << endl;

cout << endl;

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

vector<BankClient> clients; // Вектор для хранения клиентов

int choice;

do

{

cout << "Меню:" << endl;

cout << "1. Добавить клиента" << endl;

cout << "2. Вывести всех клиентов" << endl;

cout << "3. Поиск клиентов по сумме на счете" << endl;

cout << "4. Удалить клиента" << endl;

cout << "0. Выход" << endl;

cout << "Выберите действие: ";

cin >> choice;

cout << endl;

switch (choice)

{

case 1: // Добавление нового клиента

{

BankClient newClient;

cout << "Введите ФИО клиента: ";

cin.ignore(); // Очистим буфер ввода

getline(cin, newClient.fullName);

int accountTypeChoice;

cout << "Выберите тип счета (0 - Сберегательный, 1 - Текущий, 2 - Депозитный, 3 - Другой): ";

cin >> accountTypeChoice;

newClient.accountType = static\_cast<AccountType>(accountTypeChoice);

cout << "Введите номер счета: ";

cin >> newClient.accountNumber;

cout << "Введите сумму на счете: ";

cin >> newClient.balance;

int day, month, year;

cout << "Введите дату последнего изменения (день месяц год): ";

cin >> day >> month >> year;

// Заполнение данных о дате последнего изменения

newClient.lastModified.day = day;

newClient.lastModified.month = month;

newClient.lastModified.year = year;

clients.push\_back(newClient); // Добавляем клиента в вектор

cout << "Клиент успешно добавлен!" << endl << endl;

break;

}

case 2: // Вывод всех клиентов

{

if (clients.empty())

{

cout << "Нет добавленных клиентов." << endl;

}

else

{

cout << "Список клиентов:" << endl;

for (const auto& client : clients)

{

displayClient(client); // Выводим информацию о каждом клиенте

}

}

break;

}

case 3: // Поиск клиентов по сумме на счете

{

double amount;

cout << "Введите сумму для поиска (<100, >100): ";

cin >> amount;

if (amount < 100)

{

cout << "Клиенты с суммой на счете меньше 100:" << endl;

for (const auto& client : clients)

{

if (client.balance < 100)

{

displayClient(client); // Выводим информацию о клиенте

}

}

}

else

{

cout << "Клиенты с суммой на счете больше или равной 100:" << endl;

for (const auto& client : clients)

{

if (client.balance >= 100)

{

displayClient(client); // Выводим информацию о клиенте

}

}

}

break;

}

case 4: // Удаление клиента

{

int index;

cout << "Введите индекс клиента для удаления: ";

cin >> index;

if (index >= 0 && index < clients.size())

{

clients.erase(clients.begin() + index); // Удаление клиента из вектора

cout << "Клиент успешно удален!" << endl;

}

else

{

cout << "Ошибка: Неверный индекс клиента." << endl;

}

break;

}

case 0: // Выход из программы

{

cout << "Выход из программы." << endl;

break;

}

default:

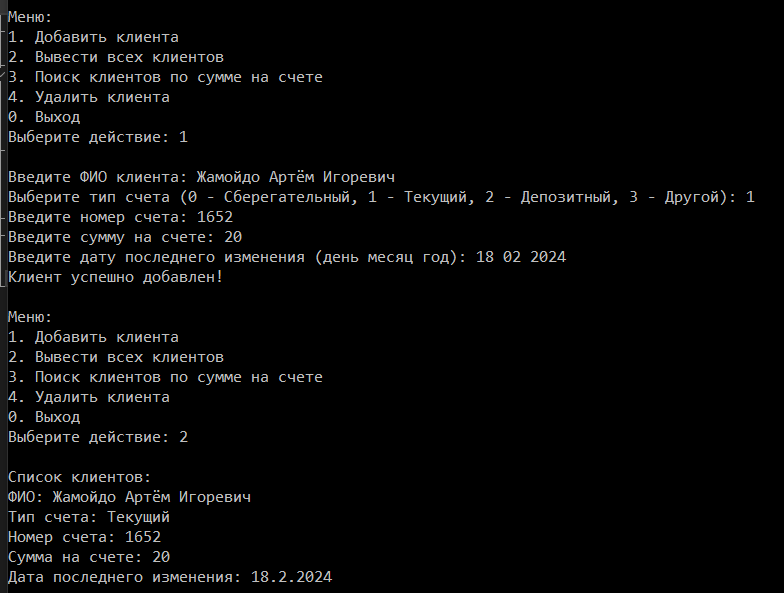
cout << "Неверный выбор. Попробуйте еще раз." << endl;

}

} while (choice != 0);

return 0;

}



В соответствии со своим вариантом разработать программу с использованием структуры в виде ***объединения****,* для работы с данными из таблицы, приведенной ниже. Реализовать функции ввода с клавиатуры, записи в файл, вывода на экран, чтения из файла и поиска.

**Клиенты банка**. Ф.И.О., тип счета (срочный, льготный и т. д.), номер счета, сумма на счете, дата последнего изменения. Выбор по номеру счета.

#include <iostream>

#include <fstream>

#include <string>

#include <Windows.h>

using namespace std;

// Объединение для хранения данных о клиенте

union ClientData

{

char fullName[100]; // Имя клиента

char accountType[20];

int accountNumber; // Номер счета

double accountBalance;

char lastModifiedDate[20]; // Дата последнего изменения

};

// Структура для удобного доступа к данным клиента

struct Client

{

ClientData data; // Данные клиента

int dataType; // Тип данных в объединении: 1 - char, 2 - int, 3 - double

};

// Функция для ввода данных о клиенте

void inputClientData(Client& client)

{

cout << "Введите данные о клиенте:\n";

cout << "1. ФИО: ";

cin.getline(client.data.fullName, 100);

cout << "2. Тип счета (1 - срочный, 2 - льготный): ";

cin.getline(client.data.accountType, 20);

cout << "3. Номер счета: ";

cin >> client.data.accountNumber;

cout << "4. Сумма на счете: ";

cin >> client.data.accountBalance;

cin.ignore();

cout << "5. Дата последнего изменения: ";

cin.getline(client.data.lastModifiedDate, 20);

client.dataType = 1; // Первоначально задаем тип данных как char

}

// Функция для вывода данных о клиенте

void displayClientData(const Client& client)

{

cout << "Данные о клиенте:\n";

cout << "ФИО: " << client.data.fullName << endl;

cout << "Тип счета: " << client.data.accountType << endl;

cout << "Номер счета: " << client.data.accountNumber << endl;

cout << "Сумма на счете: " << client.data.accountBalance << endl;

cout << "Дата последнего изменения: " << client.data.lastModifiedDate << endl;

}

// Функция для записи данных о клиенте в файл

void writeClientToFile(const Client& client, const string& fileName)

{

ofstream file(fileName, ios::app);

if (file.is\_open())

{

file << client.data.fullName << "," << client.data.accountType << ","

<< client.data.accountNumber << "," << client.data.accountBalance << ","

<< client.data.lastModifiedDate << endl;

file.close();

cout << "Данные о клиенте успешно записаны в файл." << endl;

}

else

{

cout << "Ошибка открытия файла для записи." << endl;

}

}

// Функция для чтения данных о клиентах из файла и их вывода

void readClientFromFile(const string& fileName)

{

ifstream file(fileName);

string line;

if (file.is\_open())

{

while (getline(file, line))

{

cout << line << endl;

}

file.close();

}

else

{

cout << "Ошибка открытия файла для чтения." << endl;

}

}

// Функция для поиска клиента по номеру счета

Client findClientByAccountNumber(const string& fileName, int accountNumber)

{

ifstream file(fileName);

Client client;

client.dataType = 1; // Первоначально задаем тип данных как char

while (file >> client.data.fullName >> client.data.accountType >> client.data.accountNumber

>> client.data.accountBalance >> client.data.lastModifiedDate)

{

if (client.data.accountNumber == accountNumber)

{

file.close();

return client;

}

}

file.close();

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

Client client;

string fileName = "client.txt";

int choice;

int accountNumber;

do

{

cout << "\n1. Введите данные клиента\n";

cout << "2. Вывод всех клиентов\n";

cout << "3. Запись данных в файл\n";

cout << "4. Чтение данных из файла\n";

cout << "5. Поиск клиента по номеру счета\n";

cout << "6. Выход\n";

cout << "Ваш выбор: ";

cin >> choice;

cin.ignore();

switch (choice)

{

case 1:

inputClientData(client);

break;

case 2:

displayClientData(client);

break;

case 3:

writeClientToFile(client, fileName);

break;

case 4:

readClientFromFile(fileName);

break;

case 5:

cout << "Введите номер счета для поиска: ";

cin >> accountNumber;

cin.ignore();

client = findClientByAccountNumber(fileName, accountNumber);

if (client.data.fullName[0] != '\0')

{

cout << "Клиент найден:\n";

displayClientData(client);

}

else

{

cout << "Клиент не найден.\n";

}

break;

case 6:

cout << "Конец программы.\n";

break;

default:

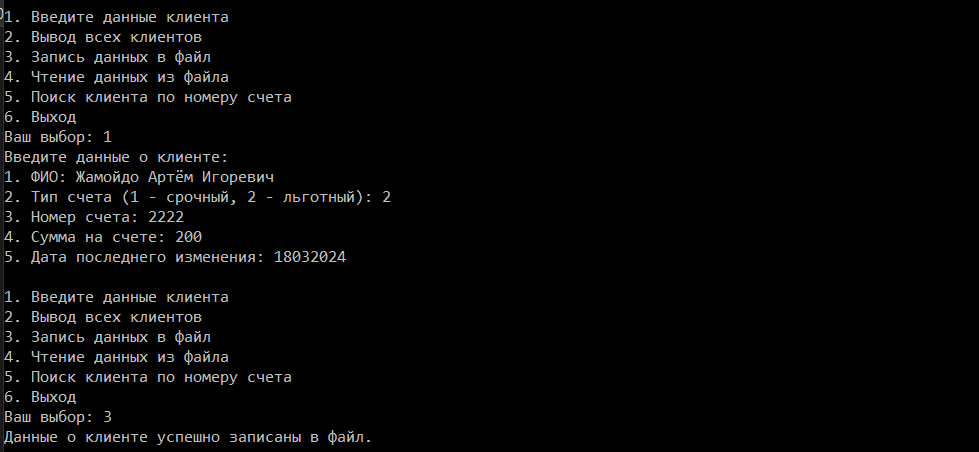
cout << "Неправильный выбор.\n";

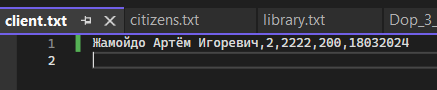
}

} while (choice != 6);

return 0;

}





Доп.задачи

**Вариант 5**

**Личная библиотека.** Автор книги, название, издательство, раздел библиотеки (специальная литература, хобби, домашнее хозяйство, беллетристика и т. д.), происхождение (покупка, кража, подарок) и наличие книги в данный момент. Выбор книг по году. Происхождение книги реализовать с помощью перечисления.

#include <iostream>

#include <string>

#include <vector>

#include <Windows.h>

using namespace std;

// Перечисление для определения происхождения книги

enum Origin

{

PURCHASE,

THEFT,

GIFT

};

// Структура для хранения информации о книге

struct Book

{

string author;

string title;

string publisher;

string section;

Origin origin;

bool available;

};

// Функция для ввода информации о книге

void inputBookData(Book& book)

{

cout << "Введите автора: ";

getline(cin, book.author);

cout << "Введите название книги: ";

getline(cin, book.title);

cout << "Введите издательство: ";

getline(cin, book.publisher);

cout << "Введите раздел библиотеки: ";

getline(cin, book.section);

cout << "Выберите происхождение книги (0 - покупка, 1 - кража, 2 - подарок): ";

int originInput;

cin >> originInput;

cin.ignore();

book.origin = static\_cast<Origin>(originInput);

cout << "Книга доступна в данный момент? (0 - Нет, 1 - Да): ";

cin >> book.available;

cin.ignore();

}

// Функция для вывода информации о книге

void displayBookData(const Book& book)

{

string originString;

switch (book.origin)

{

case PURCHASE:

originString = "Покупка";

break;

case THEFT:

originString = "Кража";

break;

case GIFT:

originString = "Подарок";

break;

default:

originString = "Неизвестно";

}

string availability = book.available ? "Да" : "Нет";

cout << "Автор: " << book.author << endl;

cout << "Название: " << book.title << endl;

cout << "Издательство: " << book.publisher << endl;

cout << "Раздел библиотеки: " << book.section << endl;

cout << "Происхождение: " << originString << endl;

cout << "Доступность: " << availability << endl;

cout << endl;

}

// Функция для удаления книги из списка

void deleteBook(vector<Book>& library, const string& title)

{

for (auto it = library.begin(); it != library.end(); ++it)

{

if (it->title == title)

{

library.erase(it);

cout << "Книга \"" << title << "\" удалена из библиотеки.\n";

return;

}

}

cout << "Книга с названием \"" << title << "\" не найдена в библиотеке.\n";

}

// Функция для поиска книги по году издания

void findBookByYear(const vector<Book>& library, int year)

{

bool found = false;

for (const Book& book : library)

{

// Предположим, что год издания можно извлечь из названия книги

if (book.title.find(to\_string(year)) != string::npos)

{

displayBookData(book);

found = true;

}

}

if (!found) {

cout << "Издания книг за " << year << " год не найдено.\n";

}

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

vector<Book> library;

int choice;

string title;

int year;

Book newBook;

do

{

cout << "\n1. Добавить книгу\n";

cout << "2. Вывести информацию о книгах в библиотеке\n";

cout << "3. Удалить книгу\n";

cout << "4. Найти книги по году издания\n";

cout << "5. Выход\n";

cout << "Ваш выбор: ";

cin >> choice;

cin.ignore();

switch (choice)

{

case 1:

inputBookData(newBook);

library.push\_back(newBook);

break;

case 2:

cout << "Информация о книгах в библиотеке:\n";

for (const Book& book : library)

{

displayBookData(book);

}

break;

case 3:

cout << "Введите название книги для удаления: ";

getline(cin, title);

deleteBook(library, title);

break;

case 4:

cout << "Введите год издания книг: ";

cin >> year;

findBookByYear(library, year);

break;

case 5:

cout << "Завершение работы.\n";

break;

default:

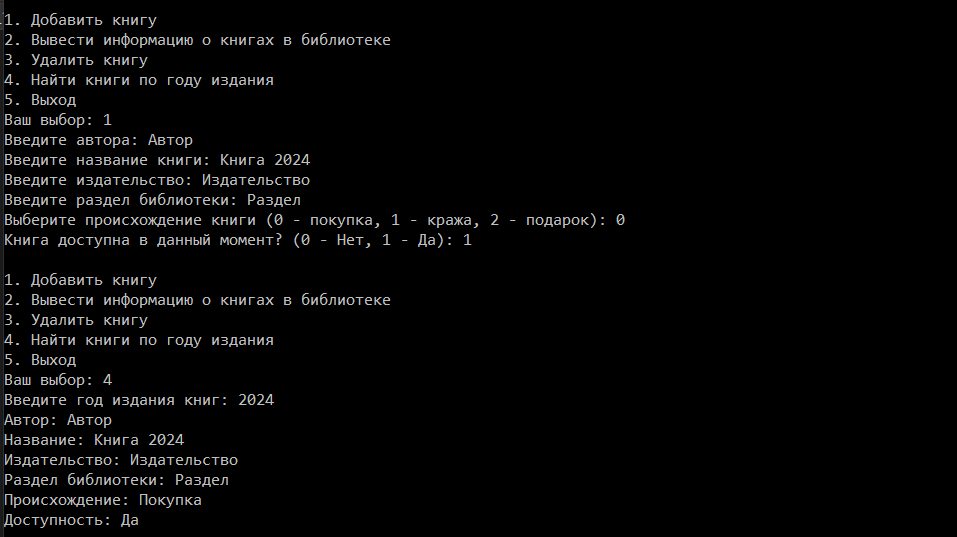
cout << "Некорректный выбор.\n";

}

} while (choice != 5);

return 0;

}

****

**Личная библиотека.** Автор книги, название, издательство, раздел библиотеки (специальная литература, хобби, домашнее хозяйство, беллетристика и т. д.), происхождение (покупка, кража, подарок) и наличие книги в данный момент. Выбор книг по автору.

#include <iostream>

#include <fstream>

#include <string>

#include <vector>

#include <Windows.h>

using namespace std;

// Перечисление для определения происхождения книги

enum Origin

{

PURCHASE,

THEFT,

GIFT

};

// Структура для хранения информации о книге

struct Book

{

// Объединение для хранения данных о книге

union

{

char author[100];

char title[100];

char publisher[100];

char section[100];

};

Origin origin; // Поле для хранения происхождения книги

bool available; // Доступность книги

};

// Функция для ввода информации о книге с клавиатуры

void inputBookDataFromConsole(Book& book)

{

cout << "Введите автора: ";

cin.getline(book.author, sizeof(book.author));

cout << "Введите название книги: ";

cin.getline(book.title, sizeof(book.title));

cout << "Введите издательство: ";

cin.getline(book.publisher, sizeof(book.publisher));

cout << "Введите раздел библиотеки: ";

cin.getline(book.section, sizeof(book.section));

int originInput;

cout << "Выберите происхождение книги (0 - покупка, 1 - кража, 2 - подарок): ";

cin >> originInput;

cin.ignore();

book.origin = static\_cast<Origin>(originInput);

cout << "Книга доступна в данный момент? (0 - Нет, 1 - Да): ";

cin >> book.available;

cin.ignore();

}

// Функция для вывода информации о книге

void displayBookData(const Book& book)

{

string originString;

switch (book.origin)

{

case PURCHASE:

originString = "Покупка";

break;

case THEFT:

originString = "Кража";

break;

case GIFT:

originString = "Подарок";

break;

}

string availability = book.available ? "Да" : "Нет";

cout << "Автор: " << book.author << endl;

cout << "Название: " << book.title << endl;

cout << "Издательство: " << book.publisher << endl;

cout << "Раздел библиотеки: " << book.section << endl;

cout << "Происхождение: " << originString << endl;

cout << "Доступность: " << availability << endl;

cout << endl;

}

// Функция для записи библиотеки в файл

void writeLibraryToFile(const vector<Book>& library, const string& filename)

{

ofstream file(filename);

if (file.is\_open())

{

for (const auto& book : library)

{

file << book.author << ',' << book.title << ',' << book.publisher << ',' << book.section << ',' << book.origin << ',' << book.available << '\n';

}

file.close();

cout << "Библиотека успешно записана в файл." << endl;

}

else

{

cout << "Ошибка открытия файла для записи." << endl;

}

}

// Функция для чтения библиотеки из файла

void readLibraryFromFile(vector<Book>& library, const string& filename)

{

ifstream file(filename);

if (file.is\_open())

{

string originString;

Book book{};

while (file.getline(book.author, sizeof(book.author), ',') &&

file.getline(book.title, sizeof(book.title), ',') &&

file.getline(book.publisher, sizeof(book.publisher), ',') &&

file.getline(book.section, sizeof(book.section), ',') &&

file >> originString &&

file >> book.available)

{

library.push\_back(book);

}

file.close();

cout << "Библиотека успешно загружена из файла." << endl;

}

else

{

cout << "Ошибка открытия файла для чтения." << endl;

}

}

// Функция для поиска книг по автору

void searchBooksByAuthor(const vector<Book>& library, const string& author)

{

cout << "Результаты поиска по автору '" << author << "':" << endl;

for (const Book& book : library)

{

if (book.author == author)

{

displayBookData(book);

}

}

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

vector<Book> library;

Book book;

string filename = "library.txt";

int choice;

string authorToSearch;

// Чтение библиотеки из файла при запуске программы

readLibraryFromFile(library, filename);

do

{

cout << "Меню:" << endl;

cout << "1. Добавить книгу" << endl;

cout << "2. Найти книги по автору" << endl;

cout << "3. Сохранить библиотеку в файл" << endl;

cout << "4. Выйти" << endl;

cout << "Выберите действие: ";

cin >> choice;

cin.ignore();

switch (choice)

{

case 1:

inputBookDataFromConsole(book);

library.push\_back(book);

break;

case 2:

cout << "Введите автора книги для поиска: ";

getline(cin, authorToSearch);

searchBooksByAuthor(library, authorToSearch);

break;

case 3:

writeLibraryToFile(library, filename);

break;

case 4:

cout << "Выход из программы." << endl;

break;

default:

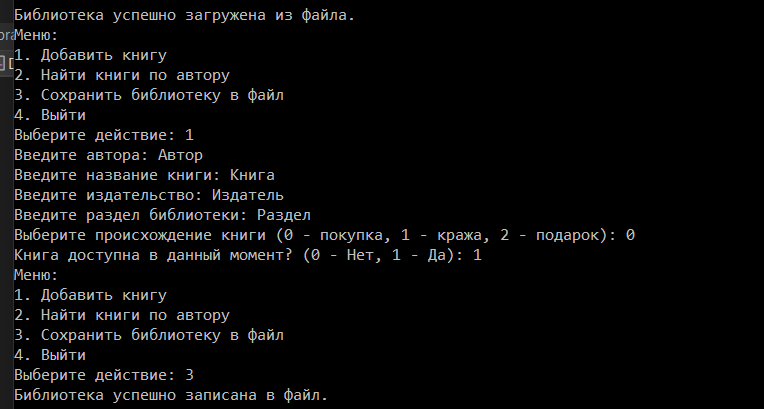
cout << "Некорректный выбор. Попробуйте снова." << endl;

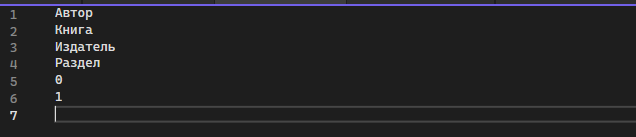
}

} while (choice != 4);

return 0;

}





**Вариант 10**

**Государство**. Наименование, столица, численность населения, площадь. Выбор государства по столице. Форму правления реализовать с помощью перечисления.

#include <iostream>

#include <string>

#include <vector>

#include <Windows.h>

using namespace std;

// Перечисление для формы правления

enum Governance

{

MONARCHY = 1,

REPUBLIC,

DICTATORSHIP,

THEOCRACY,

DEMOCRACY

};

// Структура для хранения информации о государстве

struct Country

{

string name;

string capital;

long population;

double area;

Governance governance;

};

// Функция для ввода данных о новом государстве

Country inputCountryData()

{

Country country;

cout << "Введите название государства: ";

getline(cin, country.name);

cout << "Введите название столицы: ";

getline(cin, country.capital);

cout << "Введите численность населения: ";

cin >> country.population;

cout << "Введите площадь: ";

cin >> country.area;

int governanceChoice;

cout << "Выберите форму правления (1 - Монархия, 2 - Республика, 3 - Диктатура, 4 - Теократия, 5 - Демократия): ";

cin >> governanceChoice;

country.governance = static\_cast<Governance>(governanceChoice);

cin.ignore(); // Сброс символа новой строки

return country;

}

// Функция для вывода информации о государстве

void displayCountryData(const Country& country)

{

string governanceString;

switch (country.governance)

{

case MONARCHY:

governanceString = "Монархия";

break;

case REPUBLIC:

governanceString = "Республика";

break;

case DICTATORSHIP:

governanceString = "Диктатура";

break;

case THEOCRACY:

governanceString = "Теократия";

break;

case DEMOCRACY:

governanceString = "Демократия";

break;

default:

governanceString = "Неизвестно";

}

cout << "Название государства: " << country.name << endl;

cout << "Столица: " << country.capital << endl;

cout << "Численность населения: " << country.population << endl;

cout << "Площадь: " << country.area << endl;

cout << "Форма правления: " << governanceString << endl;

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

vector<Country> countries;

int choice;

do

{

cout << "Меню:" << endl;

cout << "1. Добавить государство" << endl;

cout << "2. Вывести все государства" << endl;

cout << "3. Удалить государство" << endl;

cout << "4. Поиск государства по столице" << endl;

cout << "5. Выход" << endl;

cout << "Выберите действие: ";

cin >> choice;

cin.ignore(); // Сбрасываем символ новой строки

switch (choice)

{

case 1:

countries.push\_back(inputCountryData());

break;

case 2:

cout << "Все государства:" << endl;

for (const auto& country : countries)

{

displayCountryData(country);

cout << endl;

}

break;

case 3:

if (!countries.empty())

{

int index;

cout << "Выберите государство для удаления (от 1 до " << countries.size() << "): ";

cin >> index;

if (index >= 1 && index <= countries.size())

{

countries.erase(countries.begin() + index - 1);

cout << "Государство удалено." << endl;

}

else

{

cout << "Некорректный индекс." << endl;

}

}

else

{

cout << "Список государств пуст." << endl;

}

break;

case 4:

if (!countries.empty())

{

string capital;

cout << "Введите название столицы для поиска: ";

getline(cin, capital);

bool found = false;

for (const auto& country : countries)

{

if (country.capital == capital)

{

displayCountryData(country);

found = true;

}

}

if (!found)

{

cout << "Государство с такой столицей не найдено." << endl;

}

}

else

{

cout << "Список государств пуст." << endl;

}

break;

case 5:

cout << "Выход из программы." << endl;

break;

default:

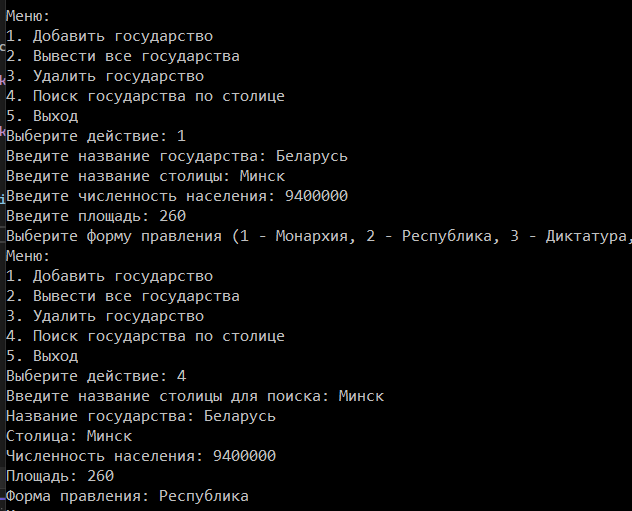
cout << "Некорректный выбор. Попробуйте снова." << endl;

}

} while (choice != 5);

return 0;

}



**Горожанин.** Ф.И.О., дата рождения, адрес, пол (м, ж). Реализовать выборку по году рождения.

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <fstream>

#include <string>

#include <vector>

#include <Windows.h>

using namespace std;

// Объединение для хранения информации о горожанине

union CitizenInfo

{

char fullName[100];

char dateOfBirth[11]; // ДД.ММ.ГГГГ

char address[100];

char gender; // 'м' - мужской, 'ж' - женский

};

// Структура с перечислением для метки типа данных в объединении

struct Citizen

{

enum Type

{

NAME,

BIRTH\_DATE,

ADDRESS,

GENDER

}

type;

CitizenInfo info;

};

// Функция для ввода данных о новом горожанине

Citizen inputCitizenData()

{

Citizen citizen{};

cout << "Введите Ф.И.О. горожанина: ";

citizen.type = Citizen::NAME;

cin.getline(citizen.info.fullName, 100);

cout << "Введите дату рождения (ДД.ММ.ГГГГ): ";

citizen.type = Citizen::BIRTH\_DATE;

cin.getline(citizen.info.dateOfBirth, 11);

cout << "Введите адрес горожанина: ";

citizen.type = Citizen::ADDRESS;

cin.getline(citizen.info.address, 100);

cout << "Введите пол горожанина (м - мужской, ж - женский): ";

citizen.type = Citizen::GENDER;

cin >> citizen.info.gender;

cin.ignore(); // Clear newline from input buffer

return citizen;

}

// Функция для вывода информации о горожанине

void displayCitizenData(const Citizen& citizen)

{

switch (citizen.type)

{

case Citizen::NAME:

cout << "Ф.И.О.: " << citizen.info.fullName << endl;

break;

case Citizen::BIRTH\_DATE:

cout << "Дата рождения: " << citizen.info.dateOfBirth << endl;

break;

case Citizen::ADDRESS:

cout << "Адрес: " << citizen.info.address << endl;

break;

case Citizen::GENDER:

cout << "Пол: " << (citizen.info.gender == 'м' ? "мужской" : "женский") << endl;

break;

}

}

// Функция для записи данных горожан в файл

void writeCitizensToFile(const vector<Citizen>& citizens, const string& filename)

{

ofstream file(filename);

if (file.is\_open())

{

for (const auto& citizen : citizens)

{

switch (citizen.type)

{

case Citizen::NAME:

file << citizen.info.fullName;

break;

case Citizen::BIRTH\_DATE:

file << citizen.info.dateOfBirth;

break;

case Citizen::ADDRESS:

file << citizen.info.address;

break;

case Citizen::GENDER:

file << citizen.info.gender;

break;

}

file << endl;

}

file.close();

cout << "Данные успешно записаны в файл." << endl;

}

else

{

cout << "Не удалось открыть файл для записи." << endl;

}

}

// Функция для чтения данных горожан из файла

vector<Citizen> readCitizensFromFile(const string& filename)

{

vector<Citizen> citizens;

ifstream file(filename);

if (file.is\_open())

{

string line;

while (getline(file, line))

{

Citizen citizen{};

size\_t pos = 0;

pos = line.find(',');

citizen.type = Citizen::NAME;

strcpy(citizen.info.fullName, line.substr(0, pos).c\_str());

line.erase(0, pos + 1);

pos = line.find(',');

citizen.type = Citizen::BIRTH\_DATE;

strcpy(citizen.info.dateOfBirth, line.substr(0, pos).c\_str());

line.erase(0, pos + 1);

pos = line.find(',');

citizen.type = Citizen::ADDRESS;

strcpy(citizen.info.address, line.substr(0, pos).c\_str());

line.erase(0, pos + 1);

citizen.type = Citizen::GENDER;

citizen.info.gender = line[0];

citizens.push\_back(citizen);

}

file.close();

}

else

{

cout << "Не удалось открыть файл для чтения." << endl;

}

return citizens;

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

vector<Citizen> citizens;

int choice;

string yearOfBirth;

do

{

cout << "Меню:" << endl;

cout << "1. Добавить горожанина" << endl;

cout << "2. Вывести всех горожан" << endl;

cout << "3. Записать горожан в файл" << endl;

cout << "4. Прочитать горожан из файла" << endl;

cout << "5. Поиск горожан по году рождения" << endl;

cout << "6. Выход" << endl;

cout << "Выберите действие: ";

cin >> choice;

cin.ignore(); // Сбрасываем символ новой строки

switch (choice)

{

case 1:

citizens.push\_back(inputCitizenData());

break;

case 2:

cout << "Все горожане:" << endl;

for (const auto& citizen : citizens)

{

displayCitizenData(citizen);

cout << endl;

}

break;

case 3:

writeCitizensToFile(citizens, "citizens.txt");

break;

case 4:

citizens = readCitizensFromFile("citizens.txt");

break;

case 5:

cout << "Введите год рождения для поиска: ";

cin >> yearOfBirth;

for (const auto& citizen : citizens)

{

if (string(citizen.info.dateOfBirth).find(yearOfBirth) != string::npos)

{

displayCitizenData(citizen);

cout << endl;

}

}

break;

case 6:

cout << "Выход из программы." << endl;

break;

default:

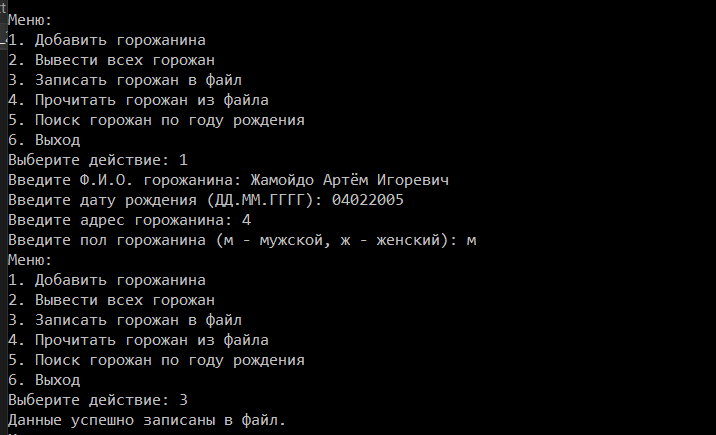
cout << "Некорректный выбор. Попробуйте снова." << endl;

}

} while (choice != 6);

return 0;

}





**Вариант 12**

**Записная книжка.** Ф.И.О, дата рождения, адрес, телефон . Поиск по фамилии. Дату рождения реализовать с помощью битового поля.

#include <iostream>

#include <string>

#include <vector>

#include <Windows.h>

using namespace std;

// Перечисление для представления даты рождения как битового поля

struct DateOfBirth

{

unsigned int day : 5;

unsigned int month : 4;

unsigned int year : 12;

};

// Структура для хранения информации о контакте

struct Contact

{

string fullName;

DateOfBirth dateOfBirth;

string address;

string phone;

};

// Функция для ввода данных о новом контакте

Contact inputContactData()

{

Contact contact;

cout << "Введите Ф.И.О. контакта: ";

getline(cin, contact.fullName);

int day, month, year;

cout << "Введите дату рождения (ДД ММ ГГГГ): ";

cin >> day >> month >> year;

contact.dateOfBirth.day = day;

contact.dateOfBirth.month = month;

contact.dateOfBirth.year = year;

cin.ignore(); // Очистка символа новой строки из ввода

cout << "Введите адрес контакта: ";

getline(cin, contact.address);

cout << "Введите номер телефона контакта: ";

getline(cin, contact.phone);

return contact;

}

// Функция для вывода информации о контакте

void displayContactData(const Contact& contact)

{

cout << "Ф.И.О.: " << contact.fullName << endl;

cout << "Дата рождения: " << contact.dateOfBirth.day << " " << contact.dateOfBirth.month << " " << contact.dateOfBirth.year << endl;

cout << "Адрес: " << contact.address << endl;

cout << "Телефон: " << contact.phone << endl;

}

// Функция для поиска контакта по фамилии

vector<Contact> searchContactsByLastName(const vector<Contact>& contacts, const string& lastName)

{

vector<Contact> foundContacts;

for (const auto& contact : contacts)

{

size\_t pos = contact.fullName.find\_last\_of(' ');

string contactLastName = contact.fullName.substr(pos + 1);

if (contactLastName == lastName)

{

foundContacts.push\_back(contact);

}

}

return foundContacts;

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

vector<Contact> contacts;

int choice;

string lastName;

do

{

cout << "Меню:" << endl;

cout << "1. Добавить контакт" << endl;

cout << "2. Вывести все контакты" << endl;

cout << "3. Поиск по фамилии" << endl;

cout << "4. Удалить контакт" << endl;

cout << "5. Выход" << endl;

cout << "Выберите действие: ";

cin >> choice;

cin.ignore(); // Очистка символа новой строки из ввода

switch (choice)

{

case 1:

contacts.push\_back(inputContactData());

break;

case 2:

cout << "Все контакты:" << endl;

for (const auto& contact : contacts)

{

displayContactData(contact);

cout << endl;

}

break;

case 3:

{

cout << "Введите фамилию для поиска: ";

getline(cin, lastName);

vector<Contact> foundContacts = searchContactsByLastName(contacts, lastName);

if (foundContacts.empty())

{

cout << "Контакты с такой фамилией не найдены." << endl;

}

else

{

cout << "Найденные контакты:" << endl;

for (const auto& contact : foundContacts)

{

displayContactData(contact);

cout << endl;

}

}

}

break;

case 4:

if (!contacts.empty())

{

int index;

cout << "Выберите контакт для удаления (введите индекс): ";

cin >> index;

contacts.erase(contacts.begin() + index); // Удаление

cout << "Контакт удален." << endl;

}

else

{

cout << "Список контактов пустой." << endl;

}

break;

case 5:

cout << "Выход из программы." << endl;

break;

default:

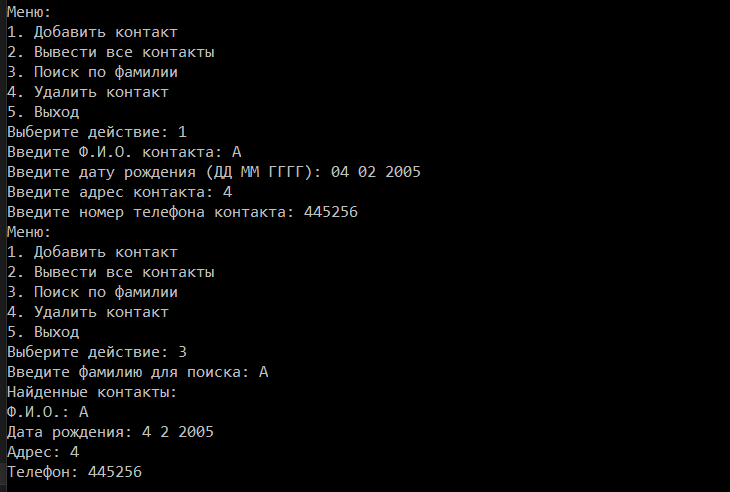
cout << "Некорректный выбор. Попробуйте снова." << endl;

}

} while (choice != 5);

return 0;

}



**Записная книжка.** Ф.И.О, дата рождения, адрес, телефон, место работы или учебы, должность. Поиск по фамилии.

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <fstream>

#include <vector>

#include <cstring>

#include <sstream>

#include <Windows.h>

using namespace std;

// Объединение для хранения информации о контакте

union ContactInfo

{

char fullName[50];

char dateOfBirth[11];

char address[100];

char phone[15];

char workplace[100];

char position[50];

};

// Структура с перечислением для метки типа данных в объединении

struct Contact

{

enum Type

{

NAME,

BIRTH\_DATE,

ADDRESS,

PHONE,

WORKPLACE,

POSITION

}

type;

ContactInfo info;

};

// Функция для ввода данных о контакте

Contact inputContactData()

{

Contact contact{};

cout << "Введите Ф.И.О. контакта: ";

contact.type = Contact::NAME;

cin.getline(contact.info.fullName, 50);

cout << "Введите дату рождения (ДД.ММ.ГГГГ): ";

contact.type = Contact::BIRTH\_DATE;

cin.getline(contact.info.dateOfBirth, 11);

cout << "Введите адрес контакта: ";

contact.type = Contact::ADDRESS;

cin.getline(contact.info.address, 100);

cout << "Введите номер телефона контакта: ";

contact.type = Contact::PHONE;

cin.getline(contact.info.phone, 15);

cout << "Введите место работы или учебы: ";

contact.type = Contact::WORKPLACE;

cin.getline(contact.info.workplace, 100);

cout << "Введите должность: ";

contact.type = Contact::POSITION;

cin.getline(contact.info.position, 50);

return contact;

}

// Функция для вывода информации о контакте

void displayContactData(const Contact& contact)

{

switch (contact.type)

{

case Contact::NAME:

cout << "Ф.И.О.: " << contact.info.fullName << endl;

break;

case Contact::BIRTH\_DATE:

cout << "Дата рождения: " << contact.info.dateOfBirth << endl;

break;

case Contact::ADDRESS:

cout << "Адрес: " << contact.info.address << endl;

break;

case Contact::PHONE:

cout << "Телефон: " << contact.info.phone << endl;

break;

case Contact::WORKPLACE:

cout << "Место работы/учебы: " << contact.info.workplace << endl;

break;

case Contact::POSITION:

cout << "Должность: " << contact.info.position << endl;

break;

}

}

// Функция для записи контакта в файл

void writeContactToFile(const Contact& contact, const string& filename)

{

ofstream file(filename, ios::app);

if (file.is\_open())

{

switch (contact.type)

{

case Contact::NAME:

file << contact.info.fullName << endl;

break;

case Contact::BIRTH\_DATE:

file << contact.info.dateOfBirth << endl;

break;

case Contact::ADDRESS:

file << contact.info.address << endl;

break;

case Contact::PHONE:

file << contact.info.phone << endl;

break;

case Contact::WORKPLACE:

file << contact.info.workplace << endl;

break;

case Contact::POSITION:

file << contact.info.position << endl;

break;

}

file.close();

}

else

{

cout << "Ошибка открытия файла для записи." << endl;

}

}

// Функция для чтения контактов из файла

vector<Contact> readContactsFromFile(const string& filename)

{

vector<Contact> contacts;

ifstream file(filename);

if (file.is\_open())

{

string line;

Contact contact{};

while (getline(file, line))

{

contact.type = static\_cast<Contact::Type>(contacts.size() % 6);

switch (contact.type)

{

case Contact::NAME:

strcpy(contact.info.fullName, line.c\_str());

break;

case Contact::BIRTH\_DATE:

strcpy(contact.info.dateOfBirth, line.c\_str());

break;

case Contact::ADDRESS:

strcpy(contact.info.address, line.c\_str());

break;

case Contact::PHONE:

strcpy(contact.info.phone, line.c\_str());

break;

case Contact::WORKPLACE:

strcpy(contact.info.workplace, line.c\_str());

break;

case Contact::POSITION:

strcpy(contact.info.position, line.c\_str());

contacts.push\_back(contact);

}

}

file.close();

}

else

{

cout << "Ошибка открытия файла для чтения." << endl;

}

return contacts;

}

// Функция для поиска контакта по фамилии

vector<Contact> searchContactsByLastName(const vector<Contact>& contacts, const char\* lastName)

{

vector<Contact> foundContacts;

for (const auto& contact : contacts)

{

// Находим фамилию в полном имени контакта

size\_t pos = strcspn(contact.info.fullName, " ");

string contactLastName = string(contact.info.fullName).substr(pos + 1);

if (strcmp(contactLastName.c\_str(), lastName) == 0)

{

foundContacts.push\_back(contact);

}

}

return foundContacts;

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

vector<Contact> contacts;

const string filename = "contacts.txt";

int choice;

do

{

cout << "Меню:" << endl;

cout << "1. Добавить контакт" << endl;

cout << "2. Вывести все контакты" << endl;

cout << "3. Записать контакты в файл" << endl;

cout << "4. Прочитать контакты из файла" << endl;

cout << "5. Поиск по фамилии" << endl;

cout << "6. Выход" << endl;

cout << "Выберите действие: ";

cin >> choice;

cin.ignore(); // Очистка символа новой строки из ввода

switch (choice)

{

case 1:

contacts.push\_back(inputContactData());

break;

case 2:

cout << "Все контакты:" << endl;

for (const auto& contact : contacts)

{

displayContactData(contact);

cout << endl;

}

break;

case 3:

for (const auto& contact : contacts)

{

writeContactToFile(contact, filename);

}

cout << "Контакты успешно записаны в файл." << endl;

break;

case 4:

contacts = readContactsFromFile(filename);

cout << "Контакты успешно прочитаны из файла." << endl;

break;

case 5:

{

char lastName[50];

cout << "Введите фамилию для поиска: ";

cin.ignore(); // Очистка символа новой строки из ввода

cin.getline(lastName, 50);

vector<Contact> foundContacts = searchContactsByLastName(contacts, lastName);

if (foundContacts.empty())

{

cout << "Контакты с такой фамилией не найдены." << endl;

}

else

{

cout << "Найденные контакты:" << endl;

for (const auto& contact : foundContacts)

{

displayContactData(contact);

cout << endl;

}

}

}

break;

case 6:

cout << "Выход из программы." << endl;

break;

default:

cout << "Некорректный выбор. Попробуйте снова." << endl;

}

} while (choice != 6);

return 0;

}

