**Тема работы:** Работа с потоками

**Цель работы:** изучить ввод-вывод в С++; научиться составлять программы с использованием функций ввода-вывода.

**Задание:**

1. Создайте программу в виде консольного приложения. Программа должна осуществлять: 1.1 Запись данных в соответствии со своим вариантом в файл на диске. Для ввода-вывода объявите тип данных – структуру. Запишите в файл 10 записей в соответствии с приведенным образцом (данные придумайте самостоятельно). 1.2 Вывод данных из файла на экран в виде таблицы с заголовком. 1.3 Производить сортировку данных в исходном файле по одному из полей и записывать полученный результат в виде таблицы в текстовый файл на диске. Поле для сортировки подчеркнуто. Выбор действия (запись, вывод, сортировка и вывод, выход из программы) реализовать в виде диалогового меню. Для этого используйте оператор switch. 2. Распечатайте полученный текстовый файл. 3. Распечатайте текст программы.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

**Ход работы**

#include <iostream>

#include <fstream>

#include <vector>

#include <string>

#include <iomanip>

#include <sstream>

#include <algorithm>

#include <stdexcept>

#include <locale>

#include <windows.h>

#include <cmath>

#include <numeric>

using namespace std;

struct Payment {

    string workshop;

    string fullName;

    int personnelNumber;

    string issueDate;

    double accrued;

    double advance;

};

// Прототипы функций

vector<Payment> loadPayments(const string& filename);

void savePayments(const string& filename, const vector<Payment>& payments);

void printTable(const vector<Payment>& payments);

void addPayment(vector<Payment>& payments);

void editPayment(vector<Payment>& payments);

void defaultData(vector<Payment>& payments);

const vector<Payment> DEFAULT\_DATA = {

    {"Доменный", "Иванов А.Н.", 145, "27.01.01", 540.00, 190.50},

    {"Мартен", "Петров С.А.", 407, "03.05.01", 730.60, 285.00},

    {"Литейный", "Сидоров В.М.", 256, "15.03.01", 620.00, 200.00},

    {"Прокатный", "Козлов И.С.", 302, "10.04.01", 680.50, 250.75},

    {"Энергетический", "Васильева О.И.", 198, "22.02.01", 590.30, 180.00},

    {"Ремонтный", "Николаев Д.К.", 431, "05.06.01", 710.25, 300.00},

    {"Транспортный", "Морозов А.А.", 523, "18.07.01", 750.00, 320.50},

    {"Инструментальный", "Зайцева Е.В.", 344, "29.08.01", 670.80, 270.30},

    {"Механический", "Белов П.Н.", 612, "14.09.01", 800.00, 350.00},

    {"Электроцех", "Григорьева М.С.", 229, "30.10.01", 650.40, 220.60}

};

string trim(const string& str) {

    size\_t first = str.find\_first\_not\_of(" \t");

    if (string::npos == first) return "";

    size\_t last = str.find\_last\_not\_of(" \t");

    return str.substr(first, (last - first + 1));

}

vector<Payment> loadPayments(const string& filename) {

    vector<Payment> payments;

    ifstream file(filename);

    if (file) {

        string line;

        while (getline(file, line)) {

            if (line.empty()) continue;

            try {

                stringstream ss(line);

                Payment p;

                getline(ss, p.workshop, ';');

                getline(ss, p.fullName, ';');

                ss >> p.personnelNumber;

                ss.ignore();

                getline(ss, p.issueDate, ';');

                ss >> p.accrued;

                ss.ignore();

                ss >> p.advance;

                payments.push\_back(p);

            }

            catch (...) {

                cerr << "Ошибка чтения строки: " << line << endl;

            }

        }

    }

    return payments;

}

void savePayments(const string& filename, const vector<Payment>& payments) {

    ofstream file(filename);

    if (!file) {

        throw runtime\_error("Ошибка открытия файла для записи");

    }

    for (const auto& p : payments) {

        file << p.workshop << ";"

            << p.fullName << ";"

            << p.personnelNumber << ";"

            << p.issueDate << ";"

            << fixed << setprecision(2) << p.accrued << ";"

            << p.advance << "\n";

    }

}

void printTable(const vector<Payment>& payments) {

    if (payments.empty()) {

        cout << "Нет данных для отображения.\n";

        return;

    }

    // Расчет ширины столбцов

    vector<size\_t> widths = { 4, 3, 15, 4, 9, 5, 10 }; // Минимальные ширины

    for (const auto& p : payments) {

        widths[0] = max(widths[0], p.workshop.length());

        widths[1] = max(widths[1], p.fullName.length());

        widths[2] = max(widths[2], to\_string(p.personnelNumber).length());

        widths[3] = max(widths[3], p.issueDate.length());

    }

    // Добавление отступов

    for (auto& w : widths) w += 2;

    // Шапка таблицы

    cout << left

        << setw(widths[0]) << "Цех"

        << setw(widths[1]) << "ФИО"

        << setw(widths[2]) << "Табельный номер"

        << setw(widths[3]) << "Дата"

        << right

        << setw(widths[4]) << "Начислено"

        << setw(widths[5]) << "Аванс"

        << setw(widths[6]) << "К выплате" << endl;

    cout << string(std::accumulate(widths.begin(), widths.end(), 0), '-') << endl;

    // Данные

    for (const auto& p : payments) {

        cout << left

            << setw(widths[0]) << p.workshop

            << setw(widths[1]) << p.fullName

            << setw(widths[2]) << p.personnelNumber

            << setw(widths[3]) << p.issueDate

            << right << fixed << setprecision(2)

            << setw(widths[4]) << p.accrued

            << setw(widths[5]) << p.advance

            << setw(widths[6]) << (p.accrued - p.advance) << endl;

    }

}

void addPayment(vector<Payment>& payments) {

    Payment p;

    cin.ignore();

    cout << "Цех: ";

    getline(cin, p.workshop);

    cout << "ФИО: ";

    getline(cin, p.fullName);

    // Проверка уникальности табельного номера

    bool unique;

    do {

        unique = true;

        cout << "Табельный номер: ";

        while (!(cin >> p.personnelNumber) || p.personnelNumber <= 0) {

            cin.clear();

            cin.ignore(10000, '\n');

            cout << "Ошибка! Введите положительное число: ";

        }

        for (const auto& payment : payments) {

            if (payment.personnelNumber == p.personnelNumber) {

                cout << "Этот номер уже существует!\n";

                unique = false;

                break;

            }

        }

    } while (!unique);

    cout << "Дата (ДД.ММ.ГГ): ";

    cin.ignore();

    getline(cin, p.issueDate);

    cout << "Начислено: ";

    while (!(cin >> p.accrued) || p.accrued < 0) {

        cin.clear();

        cin.ignore(10000, '\n');

        cout << "Ошибка! Введите неотрицательное число: ";

    }

    cout << "Аванс: ";

    while (!(cin >> p.advance) || p.advance < 0 || p.advance > p.accrued) {

        cin.clear();

        cin.ignore(10000, '\n');

        cout << "Ошибка! Введите сумму от 0 до " << p.accrued << ": ";

    }

    payments.push\_back(p);

    cout << "Запись успешно добавлена!\n";

}

void editPayment(vector<Payment>& payments) {

    if (payments.empty()) {

        cout << "Нет данных для редактирования.\n";

        return;

    }

    int number;

    cout << "Введите табельный номер для редактирования: ";

    while (!(cin >> number) || number <= 0) {

        cin.clear();

        cin.ignore(10000, '\n');

        cout << "Ошибка! Введите положительное число: ";

    }

    auto it = find\_if(payments.begin(), payments.end(),

        [number](const Payment& p) { return p.personnelNumber == number; });

    if (it == payments.end()) {

        cout << "Запись не найдена!\n";

        return;

    }

    cout << "Текущие данные:\n";

    printTable({ \*it });

    cout << "\nВведите новые данные:\n";

    Payment& p = \*it;

    cin.ignore();

    cout << "Цех [" << p.workshop << "]: ";

    string input;

    getline(cin, input);

    if (!input.empty()) p.workshop = input;

    cout << "ФИО [" << p.fullName << "]: ";

    getline(cin, input);

    if (!input.empty()) p.fullName = input;

    cout << "Дата [" << p.issueDate << "]: ";

    getline(cin, input);

    if (!input.empty()) p.issueDate = input;

    cout << "Начислено [" << p.accrued << "]: ";

    getline(cin, input);

    if (!input.empty()) p.accrued = stod(input);

    cout << "Аванс [" << p.advance << "]: ";

    getline(cin, input);

    if (!input.empty()) p.advance = stod(input);

    cout << "Данные обновлены!\n";

}

void defaultData(vector<Payment>& payments) {

    payments = DEFAULT\_DATA;

    cout << "Данные по умолчанию загружены!\n";

}

int main() {

    SetConsoleCP(1251);

    SetConsoleOutputCP(1251);

    setlocale(LC\_ALL, "Russian");

    string filename = "payments.txt";

    vector<Payment> payments;

    int choice;

    do {

        cout << "\nМеню:\n"

            << "1. Загрузить данные по умолчанию\n"

            << "2. Просмотреть данные\n"

            << "3. Добавить запись\n"

            << "4. Редактировать запись\n"

            << "5. Сохранить в файл\n"

            << "6. Загрузить из файла\n"

            << "7. Выход\n> ";

        if (!(cin >> choice)) {

            cin.clear();

            cin.ignore(10000, '\n');

            cout << "Ошибка ввода!\n";

            continue;

        }

        try {

            switch (choice) {

            case 1: defaultData(payments); break;

            case 2: printTable(payments); break;

            case 3: addPayment(payments); break;

            case 4: editPayment(payments); break;

            case 5: savePayments(filename, payments); break;

            case 6: payments = loadPayments(filename); break;

            case 7: cout << "Выход...\n"; break;

            default: cout << "Неверный выбор!\n";

            }

        }

        catch (const exception& e) {

            cerr << "Ошибка: " << e.what() << endl;

        }

    } while (choice != 7);

    return 0;

}

**Ответ:**

Изображение выглядит как текст, снимок экрана, мультимедиа, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.



Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.



**Вывод:** мы изучили ввод-вывод в С++; научиться составлять программы с использованием функций ввода-вывода.