Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный технологический университет»

ФИТ

Кафедра Информационных систем и технологий 1курс

Группа 2-2

**Структуры данных**

Тема:

Запись/чтение массива структур в файл

Лабораторная работа 8, 9

**Выполнил:**

Волковец Сергей Николаевич

**Проверил:**

Белодед Николай Иванович

Оглавление

[Теоритические сведения 3](#_Toc196858434)

[Чтение из файла 4](#_Toc196858435)

[Запись в файл 6](#_Toc196858436)

[Задания: 9](#_Toc196858437)

# Теоритические сведения

Файл - это набор данных, который хранится на компьютере или другом устройстве хранения данных. Файлы используются для хранения информации, которую можно обрабатывать или использовать позже. Они обычно имеют уникальное имя и расширение, которое указывает на тип данных, содержащихся в файле.

**Текстовые файлы** - это файлы, которые содержат символы, которые можно прочитать и интерпретировать как текст. Такие файлы, как правило, содержат данные, которые могут быть прочитаны и изменены с помощью текстовых редакторов, таких как блокнот.  
 **Двоичные файлы** - это файлы, которые содержат данные в бинарном формате, то есть данные, которые не могут быть прочитаны и поняты как текст. Такие файлы, как правило, содержат данные, которые могут быть прочитаны и обработаны только компьютером, например, изображения, звуковые файлы, исполняемые файлы и другие файлы, содержащие бинарные данные.

Двоичные файлы и текстовые файлы отличаются не только типом данных, но и методами работы с ними. Например, если мы записываем цифру «4» в текстовый файл, то она записывается как символ, и для ее хранения нужен один байт. Текстовый файл, содержащий запись: «145687», будет иметь размер шести байт.  
 Однако, если записать целое число 145 687 в двоичный файл, то он будет иметь размер четыре байта, так как именно столько необходимо для хранения данных типа int. Таким образом, двоичные файлы позволяют более эффективно использовать память компьютера, поскольку они хранят данные в бинарном формате, который занимает меньше места, чем текстовый формат.

Работа с файлами

Для работы с файлами в программировании используются специальные типы данных, называемые потоками. Поток **ifstream** служит для работы с файлами в режиме чтения, а поток **ofstream** - в режиме записи. Если нужно работать с файлом как с потоком ввода-вывода, то используется поток **fstream.**

Класс ifstream

Для чтения файлов в C++ используются потоки данных (streams). В данном случае, для работы с текстовыми файлами, используются объекты класса ifstream.

Примеры открытия файла для чтения:

1. Открытие файла с помощью метода open():

|  |
| --- |
| #include <fstream>  int main() {  std::ifstream file;  file.open("C:/file.txt"); // Открытие файла с помощью метода open()  if (file.is\_open()) { // Проверка открытия файла  // Файл успешно открыт, можно читать из него данные  }  else {  // Произошла ошибка при открытии файла  return -1;  }   file.close(); // Закрытие файла  return 0; } |

2. Открытие файла с помощью конструктора:

|  |
| --- |
| #include <fstream> using namespace std;   int main() {  ifstream file("C:/file.txt"); // Открытие файла с помощью конструктора  if (!file) { // Проверка открытия файла  // Произошла ошибка при открытии файла  return -1;  }  else {  // Файл успешно открыт, можно читать из него данные  return 1;  }  file.close(); // Закрытие файла   return 0; } |

Как показано в примере выше, для проверки успешности открытия файла, можно использовать метод is\_open(). Этот метод возвращает true, если файл успешно открыт, и false в противном случае. Также можно использовать логическое выражение с оператором "!", чтобы проверить, не равна ли переменная файла нулю. Также после работы с файлом его нужно закрыть методом close().

# Чтение из файла

Существует несколько методов чтения данных из файла:

1. Метод getline() - позволяет считать строку из файла до символа новой строки или до конца файла. Пример:

|  |
| --- |
| #include <fstream>  #include <iostream> #include <string> using namespace std;  int main() {  setlocale(LC\_ALL, "rus");  ifstream file("C:/file.txt");  if (!file.is\_open()) {  cerr << "Файл не может быть открыт!";  return 1;  }   string line;  while (getline(file, line)) {  cout << line;  }   file.close();   return 0; } |

2. Оператор >> - позволяет считывать данные из файла по одному слову или числу. Пример:

|  |
| --- |
| #include <fstream> #include <iostream> using namespace std;  int main() {  setlocale(LC\_ALL, "rus");  ifstream file("C:/file.txt");  if (!file.is\_open()) {  cerr << "Файл не может быть открыт!";  return 1;  }   int number;  while (file >> number) {  cout << number;  }   file.close();   return 0; } |

3. Метод get() - позволяет считывать данные из файла по одному символу.

|  |
| --- |
| #include <fstream> #include <iostream> using namespace std;  int main() {  setlocale(LC\_ALL, "rus");  ifstream file("C:/file.txt");  if (!file.is\_open()) {  cerr << "Файл не может быть открыт!n";     return 1;  }   char symbol;  while (file.get(symbol)) {  cout << symbol;  }  file.close();   return 0; } |

4. Метод read() - позволяет считать данные из файла в буфер заданного размера. Пример:

|  |
| --- |
| #include <fstream> #include <iostream> using namespace std;  int main() {  setlocale(LC\_ALL, "rus");  ifstream file("C:/file.txt", ios::binary);  if (!file.is\_open()) {  cerr << "Файл не может быть открыт!";   return 1;  }  char buffer[256];  while (file.read(buffer, sizeof(buffer))) {  cout.write(buffer, file.gcount());  }  file.close();   return 0; } |

Обратите внимание, что для чтения файлов в бинарном режиме необходимо указать флаг ios::binary при открытии файла.

# Запись в файл

Когда вы работаете с файлами в C++, вы можете записывать данные в файлы с помощью объектов типа ofstream. В C++ есть несколько методов, которые можно использовать для записи данных в файлы. Вот некоторые из них:

1. Метод << оператора: Этот метод записывает данные в файл с помощью перегруженного оператора <<. Этот метод может использоваться для записи простых типов данных, таких как int, double и char, а также пользовательских типов данных, которые были перегружены для работы с оператором <<.

|  |
| --- |
| #include <iostream> #include <fstream> using namespace std;  int main() {  ofstream myfile;  myfile.open("file.txt");  myfile << "Вывод в файл";  myfile << 123 << endl;  myfile.close();   return 0; } |

2. Метод write(): Этот метод записывает данные в файл, используя указатель на массив байтов и количество байтов для записи. Этот метод может использоваться для записи произвольных данных в файл. Вот пример кода:

|  |
| --- |
| #include <iostream> #include <fstream> using namespace std;  int main() {  ofstream myfile;  myfile.open("file.bin", ios::binary);  int data[] = { 1, 2, 3, 4, 5 };  int size = sizeof(data);  myfile.write((char\*)&data, size);  myfile.close();   return 0; } |

Этот код открывает файл "file.bin" для записи бинарных данных и записывает в него массив целых чисел.

3. Метод put(): Этот метод записывает один символ в файл. Этот метод может использоваться для записи символов в файл. Вот пример кода:

|  |
| --- |
| #include <iostream> #include <fstream> using namespace std;  int main() {  ofstream myfile;  myfile.open("file.txt");  myfile.put('H');  myfile.put('i');  myfile.close();  return 0; } |

Методы форматирования width, precision

Конструкция width() используется для указания минимальной ширины выводимого значения. Например, чтобы выровнять числа в колонке таблицы. Конструкция precision() используется для указания количества знаков после запятой, которые должны быть выведены для чисел с плавающей точкой. Рассмотрим пример:

|  |
| --- |
| #include <iostream> #include <iomanip> using namespace std;  int main() {  double num1 = 123.456789;  double num2 = 45.6789123;    cout << setw(10) << setprecision(5) << num1 << endl;  cout << setw(10) << setprecision(5) << num2 << endl;  return 0; } |

В этом примере мы используем setw(10), чтобы установить минимальную ширину поля в 10 символов. Мы также используем setprecision(3), чтобы указать, что мы хотим вывести только три знака.

|  |
| --- |
|  |

Режимы открытия файлов

Режимы открытия файлов устанавливают характер использования файлов. Для установки режима в классе **ios\_base** предусмотрены константы, которые определяют режим открытия файлов

|  |  |
| --- | --- |
| Константа | Описание |
| ios\_base::in | открыть файл для чтения |
| ios\_base::out | открыть файл для записи |
| ios\_base::ate | при открытии переместить указатель в конец файла |
| ios\_base::app | открыть файл для записи в конец файла |
| ios\_base::trunc | удалить содержимое файла, если он существует |
| ios\_base::binary | открытие файла в двоичном режиме |

Режимы открытия файлов можно устанавливать непосредственно при создании объекта или при вызове функции open().

|  |
| --- |
| ofstream fout("cppstudio.txt", ios\_base::app); // открываем файл для добавления информации к концу файла |

# 

# Задания:

**Задание 1.**

Реализовать возможность записи данных из структуры вашей бд в файл, а также вывод содержимого файла.

Код:

//Список клиентов гостиницы. Паспортные данные, даты приезда и отъезда, номер, тип размещения

//(люкс, одноместный, двухместный, трехместный, апартаменты).

//Поиск гостя по фамилии.

#include <iostream>

#include <string>

#include <fstream>

#include <windows.h>

using namespace std;

struct client

{

string surname;

string pasd;

string date\_join;

string date\_out;

string type\_off\_lat;

};

struct client List\_of\_hotel\_clients[100];

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

setlocale(LC\_ALL, "Russian");

int n = 0;

int choice = 1;

while (choice != 0) {

cout << "1 - ввод элементов структуры с клавиатуры\n";

cout << "2 - вывод элементов структуры в консольное окно\n";

cout << "3 - удаление заданной структурированной переменной\n";

cout << "4 - Поиск по Фамилии\n";

cout << "5 - запись информации в файл\n";

cout << "6 - чтение данных из файла.\n";

cout << "0 - Выход из программы\n\n";

cout << "Введите номер операции: ";

cin >> choice;

if (choice == 1) {

cout << "Введите данные: Фамилию, Паспортные данные, даты приезда и отъезда,тип размещения (люкс, одноместный, двухместный, трехместный, апартаменты) ";

cin >> List\_of\_hotel\_clients[n].surname;

cin >> List\_of\_hotel\_clients[n].pasd;

cin >> List\_of\_hotel\_clients[n].date\_join;

cin >> List\_of\_hotel\_clients[n].date\_out;

cin >> List\_of\_hotel\_clients[n].type\_off\_lat;

n++;

}

if (choice == 2) {

cout << "Cписок проживающих : " << endl;

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

if (List\_of\_hotel\_clients[i].surname == "") continue;

cout << List\_of\_hotel\_clients[i].surname << " ";

cout << List\_of\_hotel\_clients[i].pasd << " ";

cout << List\_of\_hotel\_clients[i].date\_join << " ";

cout << List\_of\_hotel\_clients[i].date\_out << " ";

cout << List\_of\_hotel\_clients[i].type\_off\_lat << " " << endl;

}

}

if (choice == 3) {

cout << "Введите порядковы номер человека, которого хотите удалить ";

int nn;

cin >> nn;

List\_of\_hotel\_clients[nn].surname = "";

List\_of\_hotel\_clients[nn].pasd = "";

List\_of\_hotel\_clients[nn].date\_join = "";

List\_of\_hotel\_clients[nn].date\_out = "";

List\_of\_hotel\_clients[nn].type\_off\_lat = "";

}

if (choice == 4) {

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

string ss;

cout << "Введите фамилию для поиска данных :";

cin >> ss;

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

if (List\_of\_hotel\_clients[i].surname == ss) {

cout << List\_of\_hotel\_clients[i].surname << " ";

cout << List\_of\_hotel\_clients[i].pasd << " ";

cout << List\_of\_hotel\_clients[i].date\_join << " ";

cout << List\_of\_hotel\_clients[i].date\_out << " ";

cout << List\_of\_hotel\_clients[i].type\_off\_lat << " " << endl;

}

}

}

}

if (choice == 5) {

fstream one("C:\\Users\\ASUS\\Desktop\\ЛАБЫ\\ОАИП - 2.0\\4\\1.txt");

if (!one.is\_open()) {

cout << "Ошибка при открытии файла!";

return 1;

}

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

string sss;

sss += List\_of\_hotel\_clients[i].surname + " ";

sss += List\_of\_hotel\_clients[i].pasd + " ";

sss += List\_of\_hotel\_clients[i].date\_join + " ";

sss += List\_of\_hotel\_clients[i].date\_out + " ";

sss += List\_of\_hotel\_clients[i].type\_off\_lat + " ";

one << sss << endl;

}

one.close();

cout << "Выполнено " << endl;

}

if (choice == 6) {

fstream one("C:\\Users\\ASUS\\Desktop\\ЛАБЫ\\СД\\Лабы\\4\\1.txt");

if (!one.is\_open()) {

cout << "Ошибка при открытии файла!";

return 1;

}

n = 0;

while (!one.eof()) {

one >> List\_of\_hotel\_clients[n].surname;

if (one.eof()) break;

one >> List\_of\_hotel\_clients[n].pasd;

one >> List\_of\_hotel\_clients[n].date\_join;

one >> List\_of\_hotel\_clients[n].date\_out;

one >> List\_of\_hotel\_clients[n].type\_off\_lat;

n++;

}

one.close();

cout << "Данные прочитаны из файла" << endl;

}

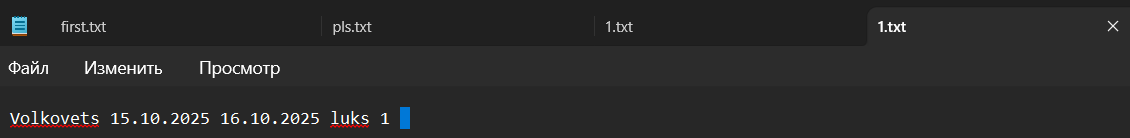
}

}

Пример работы:

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

****

**Задание 2.**

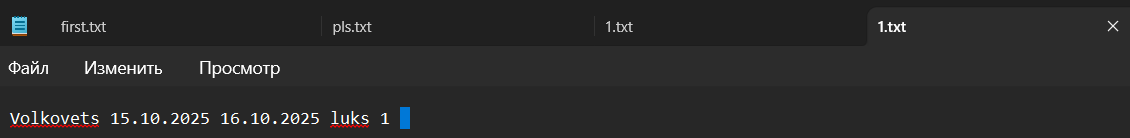
Написать функцию для считывания данных из файла в структуру вашей БД.

Код: см. задание 1

Пример работы:

**Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.**

****

**Задание 3.**

Написать программу, в которой нужно найти минимальный элемент деки и добавить его в начало, с выводом на экран .

Код:

//Написать программу, в которой нужно найти минимальный элемент

// деки и добавить его в начало, с выводом на экран .

#include <deque>

#include <iostream>

using namespace std;

int main()

{

setlocale(LC\_ALL, "Ru");

deque <double> arr = {100, 1, 0.5, 7, 8, 8.5};

double min = DBL\_MAX;

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

if (min > arr[i]) min = arr[i];

}

arr.push\_front(min);

cout << "Результат программы: ";

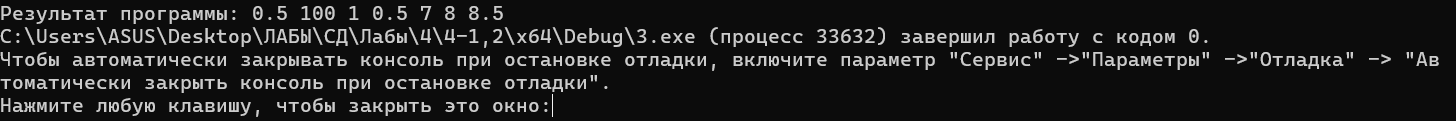
for (int i = 0; i < arr.size(); i++) {

cout << arr[i] << " ";

}

}

Пример работы кода:



**Задание 4.**

Написать программу, в которой нужно найти произведение всех элементов деки и вывести его на экран.

Код:

//Написать программу, в которой нужно найти произведение всех элементов деки и вывести его на экран.

#include <deque>

#include <iostream>

using namespace std;

int rec(const deque<int>& arr, int n) {

if (n == arr.size()) {

return 1;

}

return arr[n] \* rec(arr, n + 1);

}

int main() {

setlocale(LC\_ALL, "ru");

deque<int> arr = { 100, 2, 3, 4, 5, 6, 100 };

cout << "Произведение всех элементов деки: " << rec(arr, 0) << endl;

return 0;

}

Пример работы кода:

