Работа с файлами

Михаил Смирнов
Разработчик C++

Михаил Смирнов

О спикере:

- В C++ разработке с В C++ разработке с 2010 года
- С 2002 года работаю в Муромском Институте Владимирского Государственного Университета
- Цифровая обработка сигналов в радиолокации и гидролокации
- Траекторная обработка для радиолокаторов ближней зоны
- Создание автоматизированного рабочего места для управления гидролокатором

Вспоминаем прошлое занятие

Вопрос: что такое строка?

Вспоминаем прошлое занятие

Вопрос: что такое строка?

Ответ: строка — это массив символов

Вспоминаем прошлое занятие

Вопрос: какие типы в C++ позволяют работать со строками?

Вспоминаем прошлое занятие

Вопрос: какие типы в C++ позволяют работать со строками?

Ответ: char* и std::string

Вспоминаем прошлое занятие

Вопрос: что такое нуль-терминатор?

Вспоминаем прошлое занятие

Вопрос: что такое нуль-терминатор?

Ответ: специальный символ с кодом 0, который является частью строки и маркирует конец этой строки

Вспоминаем прошлое занятие

Вопрос: как вычислить длину строки?

Вспоминаем прошлое занятие

Вопрос: как вычислить длину строки?

Ответ: использовать функцию strlen в случае с char* или функцию length в случае с std::string

Вспоминаем прошлое занятие

Вопрос: как соединить две строки в одну?

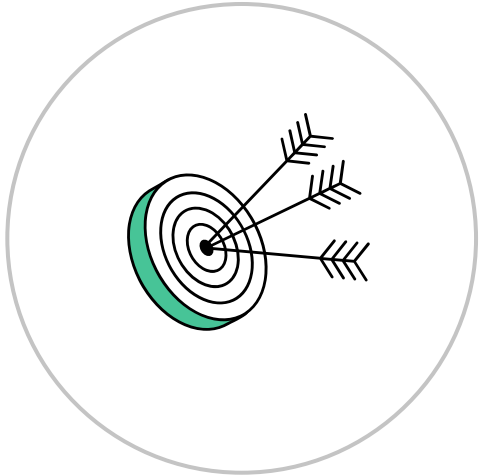
Вспоминаем прошлое занятие

Вопрос: как соединить две строки в одну?

Ответ: использовать функцию strcat
в случае с char* или оператор
+ в случае с std::string

Цели занятия

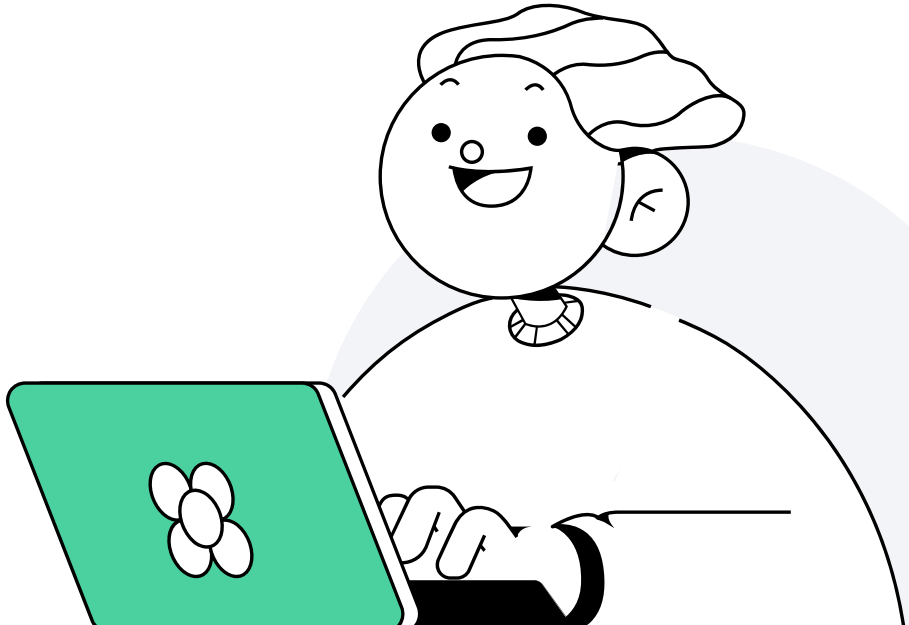
- Понять, что такое файлы
- Познакомиться с новыми типами данных для работы с файлами
- Узнать, как читать из файла
- Выяснить, как записывать в файл



План занятия

1. Что такое файлы
2. Работа с файлами
3. Итоги
4. Домашнее задание

*Нажми на нужный раздел для перехода



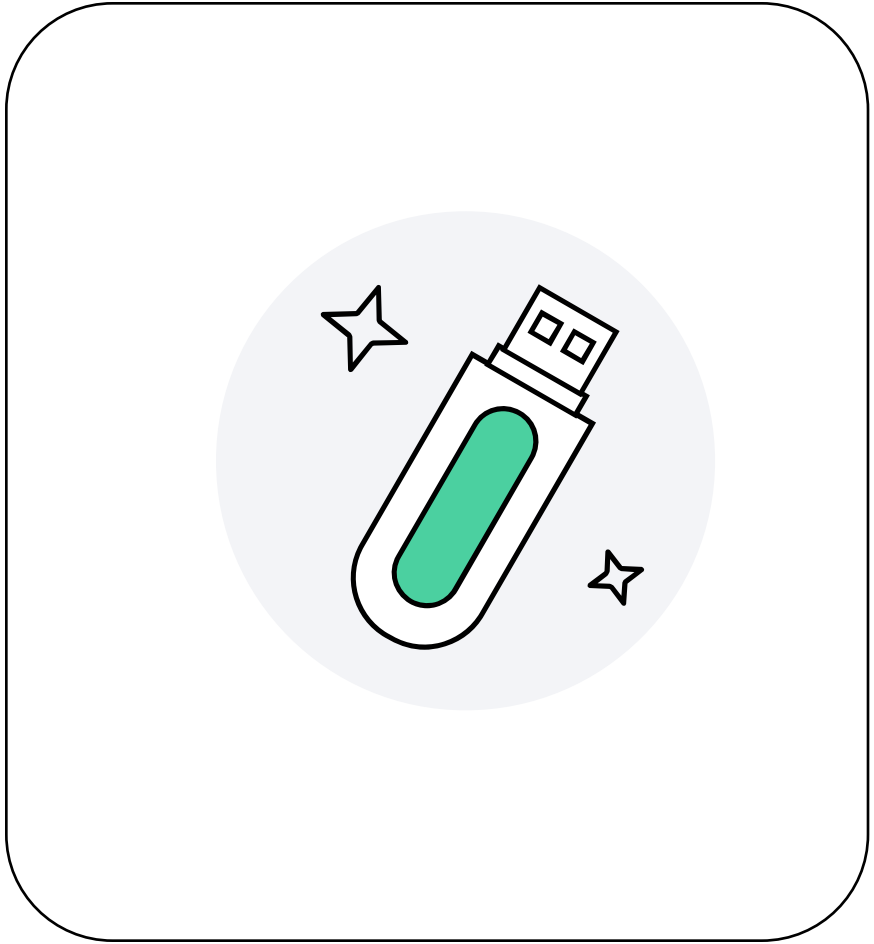
Что такое файлы

1

Что такое файлы

Файл — это именованный набор байтов, который может быть сохранён на накопителе: жёстком диске, DVD-диске, флешке.

Название файла имеет следующую структуру: **<имя файла>.<расширение>**

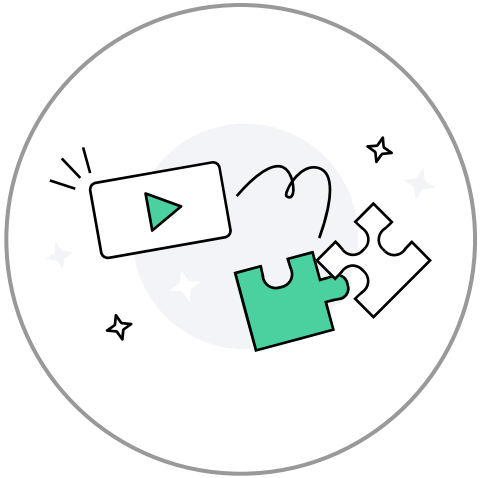


Какие бывают файлы

В рамках C++ мы для простоты будем делить файлы на два типа:

- Текстовые файлы. Содержат внутри себя текст
- Бинарные (двоичные) файлы, содержат в себе наборы байтов. Чтобы пользоваться ими, нужно знать, как их интерпретировать

На начальном уровне мы будем работать только с текстовыми файлами, которые обычно имеют расширение **.txt**

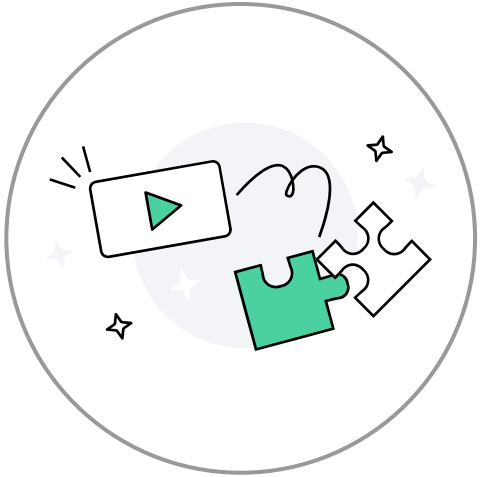


Зачем нужны файлы

Файлы используются для длительного хранения информации.

Дело в том, что запущенные на компьютере **программы** хранят свои данные, в том числе переменные, **в оперативной памяти.** Когда программа завершается, эта память освобождается для других программ, а при выключении компьютера все данные из оперативной памяти исчезают.

Файлы хранятся в постоянной памяти, поэтому при перезапусках компьютера они никуда не исчезают



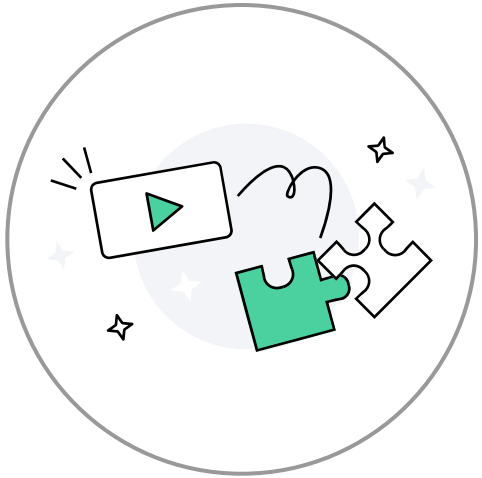
Работа с файлами

2

Работа с файлами

Работа с файлами в рамках программы в основном подразумевает:

- чтение данных из файла в программу
- запись данных из программы в файл



Типы данных

Для этих операций в C++ существует библиотека <fstream>. В ней есть несколько типов данных, которые выполняют интересующие нас задачи.

Все эти типы данных находятся в пространстве имён std:

- **std::ifstream** — для работы с файлом в режиме только на чтение
- **std::ofstream** — для работы с файлом в режиме только на запись
- **std::fstream** — для работы с файлом в режиме на чтение и на запись

Тип данных std::ifstream

Тип данных std::ifstream используется, если нужно открыть файл только для чтения. Переменная типа std::ifstream создаётся следующим образом:

std::ifstream <имя переменной> (<путь к файлу (строка)>);

```
#include <fstream>
int main(int argc, char** argv)
{
    std::ifstream fin ("C:\\data.txt"); // открыли файл C:\data.txt на чтение
}
```

Путь к файлу

Чтобы открыть файл, нужно указать путь к нему. Обратите внимание,

что в содержащей путь строке директории нужно разделять не одним символом “\” (обратный слеш), а двумя. Существует два способа, как указать путь к файлу:

- Абсолютный путь — полный путь к файлу с указанием буквы логического диска и всех промежуточных директорий. Пример: "C:\\data.txt", "D:\\Storage\\names.txt", "C:\\Users\\Admin\\Documents\\keys.txt"
- Относительный путь — путь, который рассчитывается относительно **исполняемого файла** программы, то есть .exe файла. Пример: "data.txt" попытается открыть файл data.txt в той же директории, в которой находится исполняемый файл

Чтение данных из файла

Чтение данных из файла с помощью типа данных std::ifstream очень похоже на чтение данных из консоли. Для этого используется оператор >>, который читает данные **до пробела** или **перевода строки.**

Но если при чтении данных с консоли мы использовали std::cin, то здесь вместо std::cin будем использовать созданную нами переменную типа std::ifstream:

```
#include <string>
#include <fstream>
int main(int argc, char** argv)
{
    std::string s;                      // создали переменную string для хранения строки
    std::ifstream fin ("data.txt");     // открыли файл data.txt на чтение
    fin >> s;                           // считали первое слово из файла и записали его в
                                           переменную s

}
```

Проверка на открытие

Ничто не мешает при попытке открытия файла только на чтение указать путь к файлу, которого не существует.

Узнать, получилось ли открыть файл, позволяет специальная функция, которую можно вызвать у переменной типов ifstream, ofstream и fstream: bool is_open().

Она возвращает true(1), если файл получилось открыть, и false(0), если не получилось:

```
#include <iostream>
#include <fstream>
int main(int argc, char** argv)
{
    std::ifstream fin_exist ("data.txt");                    // пытаемся открыть
                                                                существующий файл
    std::ifstream fin_not_exist ("not_existing_file.txt");   // пытаемся открыть
                                                                несуществующий файл
    std::cout << fin_exist .is_open() << std::endl;          // 1
    std::cout << fin_not_exist .is_open() << std::endl;      // 0
}
```

Как нам сделать так,
чтобы мы читали из файла **только** в том случае,
если файл смог открыться?

Напишите в чат

Ответ

Нужно использовать функцию is_open и условный оператор:

```
#include <iostream>
#include <fstream>
int main(int argc, char** argv)
{
    std::ifstream fin ("data.txt"); // пытаемся открыть файл
    if(fin.is_open())
    {
        // читаем из файла
    }
    else
    {
        std::cout << "Не получилось открыть файл!" << std::endl;
    }
}
```

Проверка на конец файла

Если мы не знаем заранее, сколько слов записано в файле, мы должны читать содержимое файла до тех пор, пока он не закончится.

Узнать, что файл закончился, позволяет функция eof(), которую можно вызвать у переменной файла. Она возвращает true, если файл уже закончился и читать больше нечего, и false, если он ещё не закончился:

```
int main(int argc, char** argv)
{
    std::string s;
    std::ifstream fin ("data.txt");       // открываем файл
    std::cout << fin.eof() << std::endl;  // 0
    fin >> s;                             // в нашем файле всего одно слово
    std::cout << fin.eof() << std::endl;  // 1
}
```

Как прочитать весь файл целиком
по одному слову за раз и вывести каждое слово
на экран на новой строке,
при этом не зная заранее, сколько в нём слов?

Напишите в чат

Ответ

Нужно использовать функцию eof и цикл while:

```
#include <iostream>
#include <fstream>
int main(int argc, char** argv)
{
  std::ifstream fin ("data.txt"); // открываем файл
  std::string s;

  while(!(fin >> s).eof())
  {
    std::cout << s << std::endl;
  }
}
```

Закрытие файла

После того, как вы воспользовались файлом и он стал вам не нужен, его необходимо закрыть. Для этого нужно вызвать функцию close() у переменной файла:

```
int main(int argc, char** argv)
{
    std::string s;
    std::ifstream fin ("data.txt");  // открываем файл
    fin >> s;                        // попользовались файлом
    fin.close();                     // закрываем файл
}
```

Тип данных std::ofstream

Тип данных std::ofstream используется, если нужно открыть файл только на запись. Переменная типа std::ofstream создаётся следующим образом:

std::ofstream <имя переменной> (<путь к файлу (строка)>);

```
#include <fstream>
int main(int argc, char** argv)
{
    std::ofstream fout ("C:\\data.txt"); // открыли файл C:\data.txt на запись
}
```

Открытие на запись несуществующего файла

В отличие от типа std::ifstream, попытка открыть несуществующий файл на запись окажется успешной. Как вы думаете, что произойдёт?

Открытие на запись несуществующего файла

Правильно, если при создании переменной типа std::ofstream вы указали путь к несуществующему файлу, то по этому пути будет создан пустой файл с указанным названием:

```
#include <fstream>
int main(int argc, char** argv)
{
    std::ofstream fout ("C:\\not_existing.txt");    // файла C:\not_existing.txt
                                                       не существовало до этой строчки
}
```

Открытие на запись несуществующего файла

Как вы думаете, если открыть существующий файл на запись и начать в него записывать, куда будет записываться новая информация? В какую часть файла?

Открытие на запись несуществующего файла

По умолчанию при открытии на запись существующего файла его старое содержимое будет **удалено**. То есть если вы откроете на запись файл с каким-то содержимым, то оно будет потеряно.

Этого можно избежать, но в этой лекции мы не будем рассматривать, как это сделать

Запись в файл

Запись в файл с использованием класса std::ofstream производится практически так же, как и вывод на консоль — с помощью оператора <<. Однако, как и в случае с типом std::ifstream, вместо std::cout используется созданная нами переменная типа std::ofstream:

```
#include <fstream>
int main(int argc, char** argv)
{
    std::ofstream fout ("C:\\data.txt");  // открыли файл C:\data.txt на запись
    fout << "Hello!" << std::endl;        // записали в файл C:\data.txt строку Hello
    fout.close();                         // закрыли файл
}
```

Тип данных std::fstream

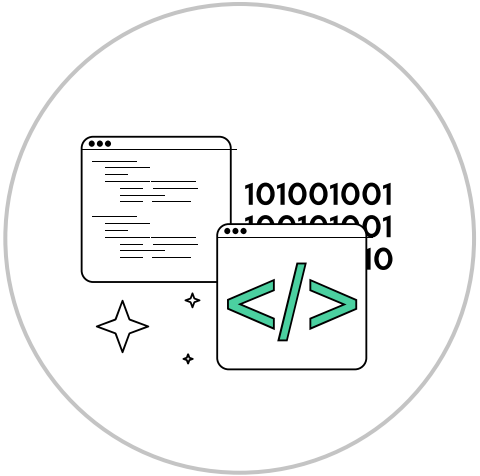
Тип данных std::fstream используется, если нужно открыть файл на чтение и на запись. Переменная типа std::fstream создаётся следующим образом:

std::fstream <имя переменной> (<путь к файлу (строка)>);

```
#include <fstream>
int main(int argc, char** argv)
{
    std::fstream f ("C:\\data.txt"); // открыли файл  C:\data.txt  на чтение и запись
}
```

Использование std::fstream

До тех пор, пока мы не умеем открывать файл на запись с сохранением его предыдущего содержимого, особого смысла в использовании типа std::fstream нет, потому что при открытии существующего файла он будет пуст



Напишем программу, которая читает данные из одного файла, записывает в другой и выводит на консоль каждое слово на новой строке

Готовый пример кода

Итоги

3

Итоги занятия

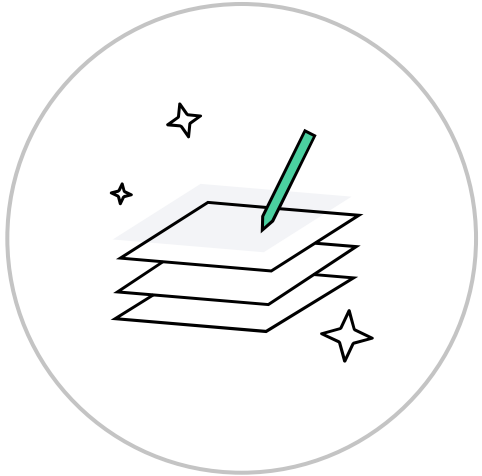
Сегодня мы

1. Поняли, что такое файлы
2. Познакомились с новыми типами данных для работы с файлами
3. Узнали, как читать из файла
4. Выяснили, как записывать в файл

Домашнее задание

Давайте посмотрим ваше домашнее задание

1. Вопросы по домашней работе задавайте в чате группы
2. Задачи можно сдавать по частям
3. Зачёт по домашней работе ставят после того, как приняты все задачи



Дополнительные материалы

- Работа с файлами

Задавайте вопросы и пишите отзыв о лекции

Михаил Смирнов
Разработчик C++