Лекция 14. Обработка исключений и работа с файлами

Тема: try/catch, собственные исключения, потоки ввода/вывода, работа с файлами (fstream).

---

1. Введение

В процессе выполнения программ могут возникать различные ошибки: деление на ноль, выход за границы массива, отсутствие файла, ошибки при вводе данных и т.д.
В традиционном программировании такие ошибки обрабатывались с помощью кодов возврата и проверок условий, что часто делало код громоздким и нечитабельным.

Язык **C++** предоставляет встроенный механизм обработки ошибок — **исключения (exceptions)**.
Он позволяет разделить основной алгоритм и обработку ошибок, делая программы более надёжными и устойчивыми.

---

2. Понятие исключения

Исключение — это событие, которое происходит во время выполнения программы и нарушает её нормальный ход.
Когда возникает исключение, управление передаётся в специальный блок, который знает, как его обработать.

Механизм обработки исключений включает три ключевых элемента:

1. **Генерация (throw)** — выбрасывание исключения;
2. **Перехват (catch)** — обработка исключения;
3. **Блок try** — область кода, где возможна ошибка.

---

3. Основной синтаксис try / catch

```
try {
    // Код, который может вызвать исключение
    int a, b;
    cin >> a >> b;
    if (b == 0)
        throw "Ошибка: деление на ноль!";
```

```
    cout << "Результат: " << a / b;
}
catch (const char* msg) {
    cout << msg << endl;
}
```

Объяснение:

- Код, который может вызвать ошибку, помещается в блок try.
- Если ошибка возникает, оператор throw «выбрасывает» исключение.
- Блок catch перехватывает это исключение и выполняет обработку.

Результат:

Введите: 10 0
Ошибка: деление на ноль!

---

4. Типы исключений

В C++ можно выбрасывать исключения различных типов:

- стандартные типы (int, string, const char*);
- объекты стандартных классов (std::exception);
- пользовательские классы.

Пример:

```
try {
    throw 404;
}
catch (int e) {
    cout << "Ошибка №" << e << endl;
}
```

---

5. Несколько блоков catch

Можно перехватывать разные типы исключений, используя несколько catch.

```
try {
    int x;
    cin >> x;
    if (x < 0) throw string("Отрицательное число!");
    if (x == 0) throw 0;
}
```

```
catch (string s) {
   cout << "Ошибка: " << s << endl;
}
catch (int n) {
   cout << "Ошибка: число равно нулю!" << endl;
}
catch (...) {
   cout << "Неизвестная ошибка!" << endl;
}
```

catch (...) — универсальный обработчик для всех типов исключений.

---

6. Собственные классы исключений

Можно определить свой класс исключений, чтобы описывать специфические ошибки программы.

```
#include <iostream>
#include <stdexcept>
using namespace std;

class DivideByZero : public runtime_error {
public:
   DivideByZero() : runtime_error("Деление на ноль!") {}
};

int main() {
   try {
      int a = 10, b = 0;
      if (b == 0)
         throw DivideByZero();
      cout << a / b;
   }
   catch (DivideByZero& e) {
      cout << "Исключение: " << e.what() << endl;
   }
}
```

Результат:

Исключение: Деление на ноль!

Таким образом, собственные классы позволяют создавать читаемую иерархию ошибок в крупных проектах.

---

7. Потоки ввода и вывода

Поток (stream) — это абстракция, представляющая источник или приёмник данных.
В C++ стандартные потоки:

Поток	Назначение
cin	стандартный ввод (клавиатура)
cout	стандартный вывод (экран)
cerr	вывод ошибок (без буферизации)
clog	лог сообщений (с буферизацией)

Пример использования:

```
int age;
cout << "Введите возраст: ";
cin >> age;
if (cin.fail()) {
   cerr << "Ошибка ввода!" << endl;
   return 1;
}
cout << "Ваш возраст: " << age << endl;
```

---

8. Работа с файлами

Файлы в C++ обрабатываются с помощью классов:

- ifstream — для чтения из файла;
- ofstream — для записи в файл;
- fstream — для чтения и записи.

Для работы необходимо подключить заголовок:

```
#include <fstream>
```

8.1. Запись данных в файл

```
#include <fstream>
#include <string>
using namespace std;
```

```
int main() {
    ofstream fout("data.txt"); // открытие файла для записи
    if (!fout) {
        cerr << "Ошибка открытия файла!" << endl;
        return 1;
    }

    fout << "Привет, мир!" << endl;
    fout << "Запись в файл выполнена успешно." << endl;
    fout.close();
}
```

После выполнения создаётся файл data.txt со строками текста.

8.2. Чтение данных из файла

```
#include <fstream>
#include <string>
#include <iostream>
using namespace std;

int main() {
    ifstream fin("data.txt");
    if (!fin) {
        cerr << "Файл не найден!" << endl;
        return 1;
    }

    string line;
    while (getline(fin, line)) {
        cout << line << endl;
    }
    fin.close();
}
```

8.3. Работа с fstream (чтение и запись)

```
#include <fstream>
using namespace std;

int main() {
    fstream file("numbers.txt", ios::in | ios::out | ios::app);
```

```
    if (!file) {
        cerr << "Ошибка открытия файла!" << endl;
        return 1;
    }

    file << "42" << endl; // добавление данных
    file.seekg(0);        // переход к началу файла

    string text;
    while (getline(file, text))
        cout << text << endl;

    file.close();
}
```

---

9. Проверка состояния потоков

После операций ввода/вывода можно проверять флаги состояния потока:

Метод	Назначение
good()	ошибок нет
eof()	достигнут конец файла
fail()	ошибка формата
bad()	критическая ошибка ввода/вывода
clear()	сброс всех флагов ошибок

Пример:

```
ifstream fin("data.txt");
if (!fin.good()) cerr << "Файл недоступен!" << endl;
```

---

10. Обработка ошибок при работе с файлами

Ошибки файловых операций можно также перехватывать с помощью исключений:

```
#include <fstream>
#include <iostream>
using namespace std;

int main() {
    ifstream file;
    file.exceptions(ifstream::failbit | ifstream::badbit);
```

```
    try {
        file.open("no_file.txt");
    }
    catch (ifstream::failure& e) {
        cerr << "Ошибка при открытии файла: " << e.what() << endl;
    }
}
```

---

11. Практический пример

```
#include <iostream>
#include <fstream>
#include <stdexcept>
using namespace std;

class FileError : public runtime_error {
public:
    FileError(const string& msg) : runtime_error(msg) {}
};

int main() {
    try {
        ifstream fin("students.txt");
        if (!fin)
            throw FileError("Файл students.txt не найден!");

        string name;
        while (getline(fin, name))
            cout << "Студент: " << name << endl;
    }
    catch (FileError& e) {
        cerr << "Ошибка: " << e.what() << endl;
    }
}
```

---

12. Заключение

Механизмы **исключений** и **работы с файлами** — важнейшие инструменты безопасного и надёжного программирования на C++.
Они позволяют:

- отделить основной алгоритм от обработки ошибок;

- предотвращать аварийные завершения программы;
- безопасно взаимодействовать с внешними данными.

Грамотное использование try/catch, пользовательских исключений и потоков fstream повышает устойчивость программ и облегчает их сопровождение.

Список литературы

1. Страуструп, Б. *Язык программирования C++*. — М.: Вильямс, 2013.
2. Лафоре, Р. *Объектно-ориентированное программирование в C++*. — СПб.: Питер, 2019.
3. Шилдт, Г. *C++ для начинающих*. — М.: Вильямс, 2020.
4. Josuttis, N. *The C++ Standard Library*. — Addison-Wesley, 2017.
5. Deitel, H. & Deitel, P. *C++ How to Program*. — Pearson, 2016.