

Технологии индустриального программирования

Лекция 3

Исключения

Р.В. Шамин

профессор кафедры индустриального программирования

Что такое исключение?

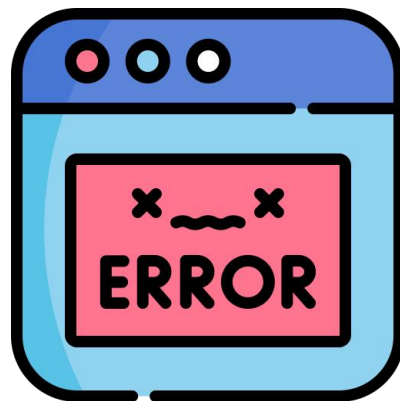
Возникающие в процессе выполнения программы ошибки или состояния, которые не позволяют дальнейшее штатное выполнение программы, называются исключениями или run-time ошибками.



Ошибки времени выполнения очень сложно отлаживать, поскольку они являются логическими, а не формальными.

Основные причины исключений:

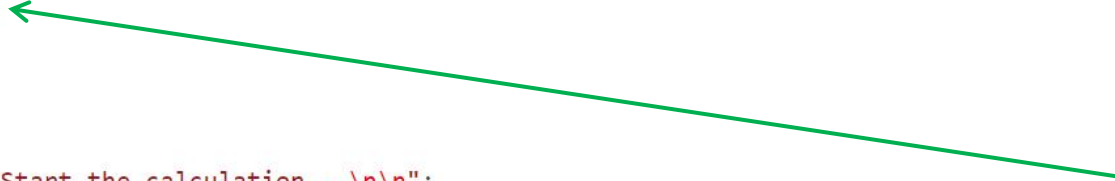
- арифметические ошибки (деление на нуль, выход за допустимую область значений функций)
- ошибки файловой системы (невозможность открыть, создать, изменить файл)
- ошибки памяти (нехватка оперативной памяти, некорректный доступ к ячейке памяти)
- сетевые ошибки (отсутствие сети и т.д.)
- программные исключения - нарушение логики программы



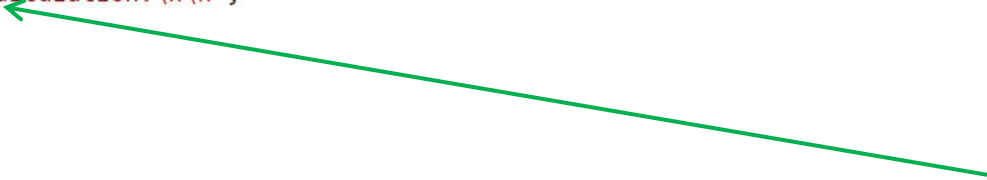
Пример исключения

```
1  #include <iostream>
2  using namespace std;
3
4  int calc(int a, int b) {
5      return a / b;
6  }
7
8  int main() {
9
10     cout << "\n\nStart the calculation...\n\n";
11
12     cout << calc(6, 0);
13
14     cout << "\n... Stop the calculation.\n\n";
15
16     return 0;
17 }
```

В этом месте выполнение программы будет прервано

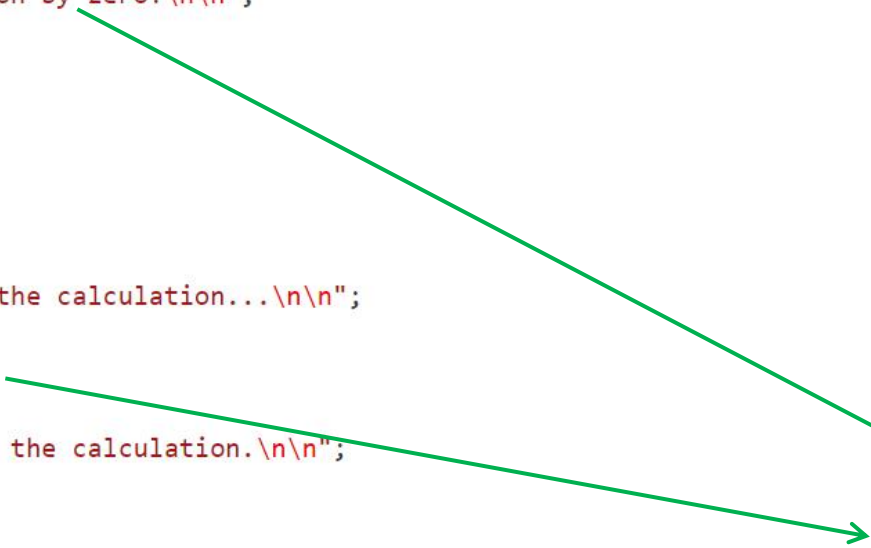


Это сообщение не будет выведено



Попытка обработать исключение

```
1  #include <iostream>
2  using namespace std;
3
4  int calc(int a, int b) {
5      if (b != 0) {
6          return a / b;
7      } else {
8          cout << "Division by zero!\n\n";
9          return 0;
10     }
11
12 }
13
14 int main() {
15     cout << "\n\nStart the calculation...\n\n";
16
17     cout << calc(6, 0);
18
19     cout << "\n... Stop the calculation.\n\n";
20
21     return 0;
22 }
23
```



```
Start the calculation...
Division by zero!
0
... Stop the calculation.
```

Обработка исключений

```
1  #include <iostream>
2  using namespace std;
3
4  int calc(int a, int b) {
5      if (b != 0) {
6          return a / b;
7      } else {
8          throw "Division by zero!";
9      }
10 }
11
12 int main() {
13     cout << "\n\nStart the calculation...\n\n";
14     try {
15         cout << calc(6, 0);
16     } catch(...) {
17         cout << "Error!\n\n";
18     }
19     cout << "\n... Stop the calculation.\n\n";
20
21     return 0;
22 }
```


Start the calculation...

Error!

... Stop the calculation.

Обработываем исключение

```
1  #include <iostream>
2  using namespace std;
3
4  int calc(int a, int b) {
5      if (b != 0) {
6          return a / b;
7      } else {
8          throw "Division by zero!";
9      }
10 }
11
12 int main() {
13     cout << "\n\nStart the calculation...\n\n";
14     try {
15         cout << calc(6, 0);
16     } catch(const char* error_message) {
17         cout << error_message << "\n\n";
18     }
19     cout << "\n... Stop the calculation.\n\n";
20
21     return 0;
22 }
```



Start the calculation...
Division by zero!
... Stop the calculation.

Генерация различных типов исключений

```
1  #include <iostream>
2  #include <cmath>
3  using namespace std;
4
5  double calc(int a, int b) {
6      double c;
7      if (b != 0) {
8          c = a / b;
9      } else {
10         throw "Division by zero!";
11     }
12     if (c < 0) {
13         throw c;
14     } else {
15         return sqrt(c);
16     }
17 }
18
19 int main() {
20     cout << "\n\nStart the calculation...\n\n";
21     try {
22         cout << calc(6, -3);
23     } catch(const char* error_message) {
24         cout << error_message << "\n\n";
25     } catch(double y) {
26         cout << "Negative: " << y << "\n\n";
27     }
28
29     cout << "\n... Stop the calculation.\n\n";
30
31     return 0;
32 }
```

Start the calculation...

Negative: -2

... Stop the calculation.

Обработка исключений и деструкторы

```
1  #include <iostream>
2  #include <cmath>
3  using namespace std;
4
5  class TCalc {
6  public:
7      double* x;
8      TCalc(double x) {
9          this->x = new double(x);
10     }
11     double run() {
12         if(*x < 0) {
13             throw *x;
14         } else {
15             return sqrt(*x);
16         }
17     }
18     ~TCalc() {
19         delete x;
20         cout << "Destructor done!";
21     }
22 };
23
24 int main() {
25
26     try {
27         TCalc Calc = TCalc(-5);
28         cout << Calc.run() << "\n\n";
29     } catch(double x) {
30         cout << "\n\nError: x = " << x << " < 0\n\n";
31     }
32
33     return 0;
34 }
```

Destructor done!

Error: x = -5 < 0

Вложенные try-catch

```
1  #include <iostream>
2  #include <cmath>
3  using namespace std;
4
5  int calc(int a, int b) {
6      if (b != 0) {
7          return a / b;
8      } else {
9          throw "Division by zero!";
10     }
11 }
12
13 int main() {
14     try {
15         try {
16             cout << calc(0, 0) << "\n\n";
17         } catch (const char* error) {
18             cout << "Inner exception: " << error << "\n\n";
19         }
20         cout << "Inner try-catch finished" << "\n\n";
21     } catch (const char* error) {
22         cout << "External exception: " << error << "\n\n";
23     }
24     cout << "External try-catch finished" << "\n\n";
25
26     return 0;
27 }
```

Inner exception: Division by zero!

Inner try-catch finished

External try-catch finished

Вложенные try-catch

```
1  #include <iostream>
2  #include <cmath>
3  using namespace std;
4
5  int calc(int a, int b) {
6      if (b != 0) {
7          return a / b;
8      } else {
9          throw "Division by zero!";
10     }
11 }
12
13 int main() {
14     try {
15         try {
16             cout << calc(0, 0) << "\n\n";
17         } catch (int error) {
18             cout << "Inner exception: " << error << "\n\n";
19         }
20         cout << "Inner try-catch finished" << "\n\n";
21     } catch (const char* error) {
22         cout << "External exception: " << error << "\n\n";
23     }
24     cout << "External try-catch finished" << "\n\n";
25
26     return 0;
27 }
```

External exception: Division by zero!

External try-catch finished

Кастомизированные исключения

```
1  #include <iostream>
2  #include <cmath>
3  using namespace std;
4
5  class TError {
6  public:
7      TError(double a, double b, double c) {
8          this->a = a;
9          this->b = b;
10         this->c = c;
11     }
12     void message() {
13         cout << "\n\nThe equation: " << a << "x^2 + " << b << "x + " << c << " = 0 has not solutions\n\n";
14     }
15 private:
16     double a, b, c;
17 };
18
19
20 int main() {
21     double a = 2, b = 3, c = 10;
22     try {
23         if ((b*b - 4.0*a*c) < 0) {
24             throw TError(a, b, c);
25         }
26     } catch (TError& error) {
27         error.message();
28     }
29
30     return 0;
31 }
```



The equation: 2x² + 3x + 10 = 0 has not solutions

Класс exception

В стандартной библиотеке определен класс exception:

```
class exception
{
public:
    exception() noexcept;
    exception(const exception&) noexcept;
    exception& operator=(const exception&) noexcept;
    virtual ~exception();
    virtual const char* what() const noexcept; // возвращает сообщение об исключении
};
```

```
1  #include <iostream>
2  #include <cmath>
3  using namespace std;
4
5  int main() {
6      double a = 2, b = 3, c = 10;
7      try {
8          if ((b*b - 4.0*a*c) < 0) {
9              throw exception();
10         }
11     } catch (exception& ex) {
12         cout << "\n\nError\n\n";
13     }
14
15     return 0;
16 }
```



Error

Класс exception

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  class TError: public exception {
6  public:
7      TError(double a, double b, double c) {
8          this->a = a;
9          this->b = b;
10         this->c = c;
11     }
12     const char* what() const noexcept override {
13         string msg = "\n\nThe equation: " + to_string(a) + "x^2 + " + to_string(b) + "x + " + to_string(c) + " = 0 has not solutions\n\n";
14         return msg.c_str();
15     }
16 private:
17     double a, b, c;
18     char buffer[255];
19 };
20
21 int main() {
22     double a = 2, b = 3, c = 10;
23     try {
24         if ((b*b - 4.0*a*c) < 0) {
25             throw TError(a, b, c);
26         }
27     } catch (exception& ex) {
28         cout << "\n\nError: " << ex.what() << "\n\n";
29     }
30
31     return 0;
32 }
```

Error:

The equation: 2.000000x^2 + 3.000000x + 10.000000 = 0 has not solutions

Стандартные обработчики исключений

runtime_error: общий тип исключений, которые возникают во время выполнения

range_error: исключение, которое возникает, когда полученный результат превосходит допустимый диапазон

overflow_error: исключение, которое возникает, если полученный результат превышает допустимый диапазон

underflow_error: исключение, которое возникает, если полученный в вычислениях результат имеет недопустимое отрицательное значение (выход за нижнюю допустимую границу значений)

logic_error: исключение, которое возникает при наличии логических ошибок в коде программы

domain_error: исключение, которое возникает, если для некоторого значения, передаваемого в функцию, не определен результат

invalid_argument: исключение, которое возникает при передаче в функцию некорректного аргумента

length_error: исключение, которое возникает при попытке создать объект большего размера, чем допустим для данного типа

out_of_range: исключение, которое возникает при попытке доступа к элементам вне допустимого диапазона

Стандартные обработчики исключений

```
1  #include <iostream>
2  using namespace std;
3
4  const int N = 1024;
5  int* A;
6
7  int main() {
8      A = new int[N];
9
10     try {
11         int n;
12         cout << "\n\nEnter n > ";
13         cin >> n;
14         if ((n < 0) || (n >= N)) {
15             throw range_error("\n\nRange error!\n\n");
16         }
17         A[n] = 10;
18     } catch (range_error& ex) {
19         cout << ex.what();
20     }
21     delete A;
22
23     return 0;
24 }
```

Enter n > 4096

Range error!