Практическое занятие 3.11

Исключения

Исключения

Исключением называется событие, которое произошло во время выполнения программы, в результате которого нормальное выполнение программы становится невозможным. Например, выход индекса за границы допустимого диапазона или другая ошибочная ситуация. В противовес исключениям применяются различные коды возврата функций и их проверка.

```
int f(int a) {
  if (a - 10 == 0)
    throw std::runtime_error
("Собрались целочисленно
делить на ноль\n");
  return 100 / (a - 10);
}
```

```
int f1(int a) {
 if (a - 10 == 0)
    return -1;
  return 100 / (a - 10);
}
```

Перехват исключений

Бросать и перехватывать можно любые типы в т.ч. простые или пользовательские.

```
float calc(float x) {
        if (x < 0) throw std::string("Sq. root of negative value");
        float res = sqrt(x);
        if (2 - res == 0) throw 0xFF;
        return 1 / (2 - res);
}
…
int main(){
  try {
      calc(2);
  }
  catch (int a) {
      std::cout <<"Error code: " << a <<"\n";
  }
  catch  (std::string st) {
      std::cout << st << "\n";
  }
}
```

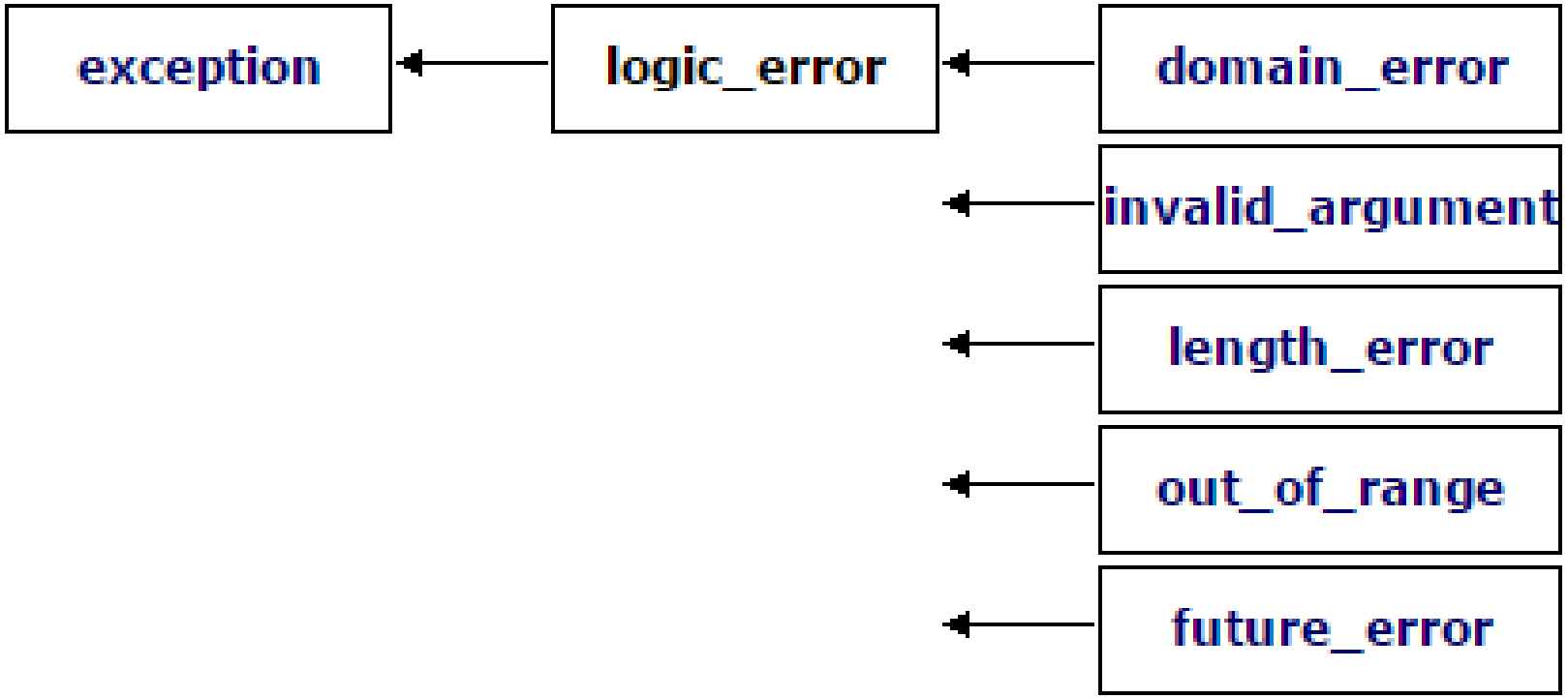
Примеры

```
/* Function 1: 1 / (2  - sqrt(x)) */
float function1(float x) {
    if(x < 0) {
        throw std::string("Square root of negative value");
    }
    float res = sqrt(x);
    if(2 - res == 0) {
        throw 0xFF;
    }
    return 1 / (2 - res);
}
/* Function 2: 1  / ((6 - x) * sqrt(x - 1)) */
float function2(int x) {
    if ((x == 6) || (x == 1)) {
        throw std::logic_error("Div by zero");
    }
    if (x < 1) {
        throw std::string("Square root of negative value");
    }
    return 1 / ((6 - x) * sqrt(x - 1));
}
```

Примеры

```
try {
        float res = function1(-1);
        std::cout << "Result 1 = " << res << "\n";
} catch (int e) {
        std::cout << "Error code: " << e << "\n";
} catch (std::string e) {
        std::cout << e << "\n";
}
try {
        float res = function2(1);
        std::cout << "Result 2 = " << res << "\n";
} catch (int e) {
        std::cout << "Error code: " << e << "\n";
} catch (std::logic_error e) {
        std::cout << e.what() << "\n";
} catch (...) {/* Ловим все возможные исключения*/
        std::cout << "ERROR!!! \n";
}
```

Стандартные исключения



Деление на ноль

```
/*This code crashes */
try {
  int h = 0;
  int a = 1 / h; }
catch(...) {
  std::cout << "Ошибка деления на ноль \n";  // ???
}
```

```
// Floating div by zero
float a1 = -1, a2 = 1, b = 0;
float c1 = a1 / b;
std::cout << "-1 / 0 = " << c1 << "\n";
float c2 = a2 / b  - 1000000;
std::cout << "1 / 0 = " << c2 << "\n";
```

Исключения в конструкторах

Если конструктор класса бросает любое исключение, память, выделенная под объект автоматически освобождается. При срабатывании исключения в конструкторе, деструктор вызван не будет.

```
Matrix(const std::vector<std::vector<int>>& _matrix) {
        int size = _matrix.at(0).size();
        for (std::vector<int> row : _matrix) {
            if (row.size() != size) {
                throw std::length_error(“Constructor row
length error");
            }
        }
```

P.S. *А в деструкторах?!*

Google Style Guide

Малоизвестная компания Google опубликовала Style guide по **C++**, там есть раздел про исключения.

ЗА исключения:

• Помогают определять поведение приложения на необходимом уровне без ведения справочника кодов ошибок.

• Повсеместно используются в других современных языках.

•Обеспечивают разделение между логикой программы и обработкой ошибок.

•Только с помощью исключений можно «аварийно», но безопасно выйти из конструктора, иначе придется писать фабрики, что несет в себе дополнительные затраты.

Google Style Guide

Против исключений:

•При добавлении исключений в существующую функцию нужно убедится, что оно обрабатывается уровнями выше.

•Множество исключений затрудняет понимание логики программы.

•При работе с исключениями нужно аккуратно смотреть за работой с памятью, при «аварийном» выходе из функций, данные в куче не очищаются автоматически.

•Исключения добавляют «веса» программам.

•Существует много ситуаций, когда исключения бросать не нужно, например, обработка ввода пользователя и др.

•Множество ранее написанных программ не использует исключения.

Вывод: **Google** не рекомендует использовать исключения;
а **Microsoft** – рекомендует -
https://msdn.microsoft.com/ru-ru/library/hh279678.aspx