Обработка исключений

» Наша библиотека

```
int ReadIntegerInRange(int min, int max)
{
   int value = 0;
   cin >> value;
   bool isCorrect = ((value > min)
               && (value < max));
   if (!isCorrect)
   {
      cout << "Повторите ввод. ";
      cin >> value;
   }
   return value;
}
```

» Наша библиотека

```
int ReadIntegerInRange(int min, int max)
{
   int value = 0;
   cin >> value;
   bool isCorrect = ((value > min)
               && (value < max));
   if (!isCorrect)
   {
      cout << "Повторите ввод. ";
      cin >> value;
   }
   return value;
}
```

» Сторонняя программа

```
int ReadCost()
{
   //Ввод стоимости товара
   int cost = 0;
   cout << "Введите стоимость товара "
        << "(0-100 000): ";
   cost = ReadIntegerInRange(500, 100);
   return cost;
}
```

» Наша библиотека

```
int ReadIntegerInRange(int min, int max)
{
   int value = 0;
   cin >> value;
   bool isCorrect = ((value > min)
               && (value < max));
   if (!isCorrect)
   {
      cout << "Повторите ввод. ";
      cin >> value;
   }
   return value;
}
```

» Сторонняя программа

```
int ReadCost()
{
   //Ввод стоимости товара
   int cost = 0;
   cout << "Введите стоимость товара "
        << "(0-100 000): ";
   cost = ReadIntegerInRange(500, 100);
   return cost;
}
```

» Наша библиотека

```
int ReadIntegerInRange(int min, int max)
{
   int value = 0;
   cin >> value;
   bool isCorrect = ((value > min)
               && (value < max));
   if (!isCorrect)
   {
      cout << "Повторите ввод. ";
      cin >> value;
   }
   return value;
}
```

» Сторонняя программа

```
int ReadCost()
{
   //Ввод стоимости товара
   int cost = 0;
   cout << "Введите стоимость товара "
        << "(0-100 000): ";
   cost = ReadIntegerInRange(500, 100);
   return cost;
}
```

Функция отработает некорректно, но программа не завершится – пользователь может не заметить, что произошла ошибка

» Наша библиотека

```
int ReadIntegerInRange(int min, int max) {
   if (min < max)
   {
      int temp = min;
      min = max;
      max = temp;
   }
   int value = 0;
   cin >> value;
   bool isCorrect = ((value > min)
               && (value < max));
   if (!isCorrect)
   {
      cout << "Повторите ввод. ";
      cin >> value;
   }
   return value;
}
```

» Сторонняя программа

```
int ReadCost()
{
   //Ввод стоимости товара
   int cost = 0;
   cout << "Введите стоимость товара "
        << "(0-100 000): ";
   cost = ReadIntegerInRange(500, 100);
   return cost;
}
```

Написание подобной перестановки – неправильно!

Другой программист совершил ошибку в коде, а мы её маскируем

В результате, программист никогда не узнает, что у него есть ошибка в коде

**Нужно писать такой код,
который не позволяет
использовать себя неправильно**

Исключения

— ошибка времени выполнения и другие возможные проблемы, которые могут возникнуть при выполнении программы и приводят к невозможности (бессмысленности) дальнейшей отработки программой её базового алгоритма:

- невозможность получения доступа к внешним данным;
- отсутствие свободной памяти для резервирования программой;
- и т.д.

Исключения

— ошибка времени выполнения и другие возможные проблемы, которые могут возникнуть при выполнении программы и приводят к невозможности (бессмысленности) дальнейшей отработки программой её базового алгоритма:

- невозможность получения доступа к внешним данным;
- отсутствие свободной памяти для резервирования программой;
- и т.д.

Общая реакция на любую исключительную ситуацию является немедленное прекращение выполнения программы

» Наша библиотека

```
#include <exception>

int ReadIntegerInRange(int min, int max) {
   if (min > max)
      throw exception("Значение min превышает
                       max");
   int value = 0;
   cin >> value;
   bool isCorrect = ((value > min)
               && (value < max));
   if (!isCorrect) {
      cout << "Повторите ввод. ";
      cin >> value;
   }
   return value;
}
```

» Сторонняя программа

```
int ReadCost()
{
   //Ввод стоимости товара
   int cost = 0;
   cout << "Введите стоимость товара "
        << "(0-100 000): ";
   cost = ReadIntegerInRange(500, 100);
   return cost;
}
```

Throw

…

`throw exception("Значение min превышает max");` - создание и выбрасывание ошибки - исключения

…

`exception` – специальный тип данных, хранящий данные об возникшей исключительной ситуации

`exception("Значение min превышает max")` – создание исключения с текстом ошибки

`throw` – ключевое слово, выбрасывающее исключение

Теперь, в случае неправильного вызова функции, программа аварийно завершится - и это хорошо!

Потому что другой разработчик, неправильно вызвавший нашу функцию, не сможет не заметить аварийного завершения программы

Ему останется только исправить свою ошибку

Частые проверки для исключений

1) Проверка входного указателя на NULL
2) Проверка входного массива на существование
3) Проверка на неотрицательность длины массива
4) Проверка, не является ли входная строка пустой
5) Проверка, не является ли входной список пустым (vector/list из STL)
6) Проверка принадлежности числа диапазону

и т.д.

Если вы не делаете проверку исключений, подпишите в комментарии к функции требования к передаваемым данным

Если вы не делаете проверку исключений, подпишите в комментарии к функции требования к передаваемым данным

```
// Массив value должен быть не пустой и не null
int IndexOf(double* values, int count, double findedValue)
{
...
}
```

Аварийное завершение программы
– не всегда желательный результат
работы программы.

Исключение может возникнуть
при неправильном действии пользователя.

В таком исключительной ситуации не требуется завершение программы, а лишь запросить повторный ввод данных пользователем или использовать значения по умолчанию.

Обработка исключений

```
int exceptions_1_2_main()
{
   int min = ReadMinFromDataBase(); // min = 200
   int max = ReadMaxFromDataBase(); // max = 100
   int value = 0;

   try
   {
      value = ReadIntegerInRange(min, max);
   }
   catch (exception e)
   {
      cout << "Произошла ошибка!" << endl
           << e.what() << endl;
      value = 15; // В случае ошибки берем значение по умолчанию
   }

   return 0;
}
```

Итог

- Ключевое слово throw – выбрасывает исключение. Если исключение не обрабатывается в программе, программа аварийно завершается.
- В блок try помещается код, в котором мы ожидаем генерации исключения.
- В блок catch помещается код, который сработает только в том случае, если в блоке try возникло исключение.

- exception – специальный тип данных, используемый в генерации исключений

Механизм обработки исключений позволяет:

1) сгенерировать ошибку в случаях неправильного использования нашего кода другими разработчиками и помочь им найти ошибки в своём исходном коде;
2) предотвратить аварийное завершение нашей программы в случае исключительной ситуации при использовании сторонних библиотек.

Литература

1) Лафоре Р. «Объектно-ориентированное программирование в Си++».
   - Глава 14 «Шаблоны и исключения»

2) Шилдт Г. «Си++ для начинающих».
   - Модуль 12 «Исключения, шаблоны и кое-что еще»

3) Исключения. Обработка исключений [Электронный ресурс]: metanit.com. Сайт о программировании // URL: https://metanit.com/cpp/tutorial/6.1.php