

Винятки throw, try i catch

```
1 int findFirstChar(const char* string, char ch)
2 {
3     // Перебираем каждый символ строки
4     for (int index=0; index < strlen(string); ++index)
5         // Если текущий символ совпадает со значением переменной ch,
6         if (string[index] == ch)
7             return index;
8
9     // Если совпадение не найдено, то возвращаем -1
10    return -1;
11 }
```

```
1 throw -1; // генерация исключения типа int
2 throw ENUM_INVALID_INDEX; // генерация исключения типа enum
3 throw "Can not take square root of negative number"; // генерация исключения типа const char*
4 throw dX; // генерация исключения типа double (переменная типа double, которая была определена
5 throw MyException("Fatal Error"); // генерация исключения с использованием объекта класса MyEx
```

```
1 try
2 {
3     // Здесь мы пишем стейтменты, которые будут генерировать следующее исключение
4     throw -1; // типичный стейтмент throw
5 }
```

```
1 catch (int a)
2 {
3     // Обрабатываем исключение типа int
4     std::cerr << "We caught an int exception with value" << a << '\n';
5 }
```

```
1 catch (double) // примечание: мы не указываем имя переменной, так как в этом нет надобности
2 {
3     // Обрабатываем исключение типа double здесь
4     std::cerr << "We caught an exception of type double" << '\n';
5 }
```

```

1  #include <iostream>
2  #include <string>
3
4  int main()
5  {
6      try
7      {
8          // Здесь мы пишем стейтменты, которые будут генерировать следующее исключение
9          throw -1; // типичный стейтмент throw
10     }
11     catch (int a)
12     {
13         // Любые исключения типа int, сгенерированные в блоке try выше, обрабатываются здесь
14         std::cerr << "We caught an int exception with value: " << a << '\n';
15     }
16     catch (double) // мы не указываем имя переменной, так как в этом нет надобности (мы её нигде не используем)
17     {
18         // Любые исключения типа double, сгенерированные в блоке try выше, обрабатываются здесь
19         std::cerr << "We caught an exception of type double" << '\n';
20     }
21     catch (const std::string &str) // ловим исключения по константной ссылке
22     {
23         // Любые исключения типа std::string, сгенерированные внутри блока try выше, обрабатываются здесь
24         std::cerr << "We caught an exception of type std::string" << '\n';
25     }
26
27     std::cout << "Continuing our way!\n";
28
29     return 0;
30 }

```

We caught an int exception with value -1
Continuing our way!

Обробка винятків, насправді, досить-таки проста, і все, що вам потрібно запам'ятати:

- При генерації виключення (оператор *throw*), точка виконання програми негайно переходить до найближчого блоку *try*. Якщо який-небудь з обробників *catch*, прикріплених до блоку *try*, обробляє цей тип виключення, то точка виконання переходить в цей обробник і, після виконання коду блоку *catch*, виняток вважається обробленим.
- Якщо відповідних обробників *catch* не існує, то виконання програми переходить в наступний блок *try*. Якщо до кінця програми не знайдені відповідні обробники *catch*, то програма завершує своє виконання з помилкою виключення.

```

1 #include <iostream>
2 #include "math.h" // для функции sqrt()
3
4 int main()
5 {
6     std::cout << "Enter a number: ";
7     double a;
8     std::cin >> a;
9
10    try // ищем исключения внутри этого блока и отправляем их в соответствующий обработчик catch
11    {
12        // Если пользователь ввёл отрицательное число, то выбрасывается исключение
13        if (a < 0.0)
14            throw "Can not take sqrt of negative number"; // выбрасывается исключение типа const char*
15
16        // Если пользователь ввёл положительное число, то выполняется операция и выводится результат
17        std::cout << "The sqrt of " << a << " is " << sqrt(a) << '\n';
18    }
19    catch (const char* exception) // обработчик исключений типа const char*
20    {
21        std::cerr << "Error: " << exception << '\n';
22    }
23 }

```

Enter a number: 16

The sqrt of 16 is 4

Enter a number: -3

Error: Can not take sqrt of negative number

```

1 #include <cmath> // для sqrt()
2 #include <iostream>
3
4 // Отдельная функция вычисления квадратного корня
5 double mySqrt(double a)
6 {
7     // Если пользователь ввёл отрицательное число, то выбрасываем исключение
8     if (a < 0.0)
9         throw "Can not take sqrt of negative number"; // выбрасывается исключение типа const char*
10
11     return sqrt(a);
12 }
13
14 int main()
15 {
16     std::cout << "Enter a number: ";
17     double a;
18     std::cin >> a;
19
20    try // ищем исключения, которые выбрасываются в блоке try и отправляем их для обработки в блок(и) catch
21    {
22        double d = mySqrt(a);
23        std::cout << "The sqrt of " << a << " is " << d << '\n';
24    }
25    catch (const char* exception) // обработка исключений типа const char*
26    {
27        std::cerr << "Error: " << exception << std::endl;
28    }
29
30    return 0;
31 }

```

Enter a number: -3

Error: Can not take sqrt of negative number

```

1 #include <iostream>
2 #include <cmath> // для sqrt()
3
4 // Отдельная функция вычисления квадратного корня числа
5 double mySqrt(double a)
6 {
7     // Если пользователь ввёл отрицательное число,
8     if (a < 0.0)
9         throw "Can not take sqrt of negative number"; // то выбрасывается исключение типа const char*
10
11     return sqrt(a);
12 }
13
14 int main()
15 {
16     std::cout << "Enter a number: ";
17     double a;
18     std::cin >> a;
19
20     // Здесь нет никакого обработчика исключений!
21     std::cout << "The sqrt of " << a << " is " << mySqrt(a) << '\n';
22
23     return 0;
24 }

```

```

1 #include <iostream>
2
3 int main()
4 {
5     try
6     {
7         throw 7; // выбрасывается исключение типа int
8     }
9     catch (double a)
10    {
11        std::cout << "We caught an exception of type double: " << a << '\n';
12    }
13    catch (...) // обработчик catch-all
14    {
15        std::cout << "We caught an exception of an undetermined type!\n";
16    }
17 }

```

We caught an exception of an undetermined type!

```
1  #include <iostream>
2
3  int main()
4  {
5
6      try
7      {
8          runGame();
9      }
10     catch(...)
11     {
12         std::cerr << "Abnormal termination\n";
13     }
14
15     saveState(); // сохраняем текущее состояние игрока
16     return 1;
17 }
```