Лекция 12

Обработка исключений

Исключительные ситуации

Исключительная ситуация (исключение, exception) — это ошибка времени выполнения программы или иная (возможно внешняя) проблема, приводящая к невозможности или бессмысленности дальнейшей отработки программой её базового алгоритма.

Примеры исключительных ситуаций:

1) деление на ноль в очередном операторе
2) отсутствие файла для чтения данных
3) блокировка записи в базе данных
4) нехватка динамической памяти
5) отключение

Виды исключений

Синхронные – возникают только в определённых, заранее известных точках программы. Примеры: ошибка чтения файла или коммуникационного канала, нехватка памяти.

Асинхронные – возникают в любой момент времени и не зависят от того, какая конкретно инструкция программы выполняется. Примеры: аварийный отказ питания; поступление новых данных в момент, когда они не ожидаются.

Обработка ошибок в языке C – функция возвращает код ошибки

```
int DoSomething()
{
    long *a = malloc(sizeof(long) * 10);
    if (a == NULL)
      return 1;
    FILE *b = fopen("mydata.txt", "rb");
    if (b == NULL)
    {
  free(a);
  return 2;
    }
}
```

Исключительные ситуации в C++: обработка исключений

Реакция программы на ошибку (исключение)

1) прекращение выполнения программы
2) попытка исправить ситуацию
3) если исправление невозможно, то выдача сообщения об ошибке

При обработке исключения программой управление передается заранее определенному **обработчику исключения** – блоку кода, выполняющему необходимые действия по минимизации потерь.

Обработка исключений в C++

Ключевые слова C++

throw (*бросать*) – генерация исключения, то есть сигнала о возникновении ошибки;

try (*пытаться*) – обозначение контролируемого блока кода, в котором возможно возникновение ошибок;

catch (*ловить*) – собственно обработка исключения, в том числе попытки исправить ошибки.

1) Генерация (выброс) исключения

Обработка исключения начинается с появления ошибки. Функция, в которой возникла ошибка, генерирует исключение.

Синтаксис

```
throw выражение;
```

Здесь выражение определяет вид исключения.

Пример:

```
if (x == 0)
    throw "деление на ноль!";
```

2) Контролируемый блок кода

Обработка исключения производится внутри **контролируемого блока**, то есть набора потенциально опасных операторов, выполнение которых может привести к ошибкам.

Синтаксис определения контролируемого блока

```
try
{
    операторы;
}
```

3) Обработчик исключения

Обработчик исключения располагается непосредственно за блоком try и начинается с ключевого слова catch. Вслед за ним в скобках указывается тип обрабатываемого исключения.

Формы записи

```
catch (тип имя) { тело_обработчика; }
catch (тип) { тело_обработчика; }
catch (...) { тело_обработчика; }
```

Пример обработки исключения

```
#include <fstream.h>

class Hello
{
   // Класс, информирующий о своем
   // создании и уничтожении
   public:
     Hello(){ cout << "Hello!" << endl; }
     ~Hello(){ cout << "Bye!" << endl; }
};
```

```
void f1() // функция, содержащая ошибку
{
   ifstream ifs("\\INVALID\\FILE\\NAME");
   if (!ifs)
   {
      cout << "Генерируем исключение";
      throw "Ошибка открытия файла";
   }
}

void f2() // функция, вызывающая f1
{
   Hello H; // Создаем объект H
   f1();    // Вызываем функцию f1
}
```

```
int main()
{
    try
    {
      cout << "Вход в try-блок" << endl;
      f2();
      cout << "Выход из try-блока" << endl;
    }
    catch(int i)
    {
      cout << "Вызван обработчик int" <<endl;
      return -1;
    }
    ...
```

```
  ...
catch(const char *p)
{
    cout << "Вызван обработчик const char*.
             \nИсключение - " << p << endl;
    return -1;
}
ca
{
}
return 0; // Все обошлось благополучно
}
```

```
Вход в try-блок
Hello!
Генерируем исключение
Bye!
Вызван обработчик const char *.
Исключение – Ошибка открытия файла
```

Последовательность событий при возникновении ошибки

1) Код выполняется стандартно вне контролируемого блока try.
2) Управление переходит в контролируемый блок (try).
3) Какое-то выражение в этом блоке приводит к возникновению ошибки.
4) Производится генерация исключения (throw).
5) Управление передается обработчику событий (catch).

Другой пример – стек

```
#include <iostream>
using namespace std;
const int MAX = 3;     // 3 элемента

class Stack
{
  private:
     int st[MAX];      // массив данных
     int top;          // вершина стека
  public:
     class Range{};
     Stack(){ top = -1; }
   ...
```

```
  void push(int var)
  {
     if(top >= MAX-1)   //если стек полон,
     throw Range();     //исключение
     st[++top] = var;   //помещаем число
  }
  //---------------------------------
  int pop()
  {
     if(top < 0)        //если стек пуст,
     throw Range();     //исключение
     return st[top--]; //извлечь число
  }
};
...
```

```
int main()
{
  Stack s1;

  try
  {
     s1.push(11);
     s1.push(22);
     s1.push(33);
     s1.push(44);     //Стек переполнен!
     cout << "1: " << s1.pop() << endl;
     cout << "2: " << s1.pop() << endl;
     cout << "3: " << s1.pop() << endl;
     cout << "4: " << s1.pop() << endl;
                      //Стек пуст!
  }
```

```
    ...
    catch(Stack::Range)   // Обработчик
    {
      cout << "Ошибка!" << endl;
    }
    cout << "Выполняется после catch" <<endl;
  return 0;
}
```