acm.mipt.ru

олимпиады по программированию на Физтехе

Раздел «Язык Си» . OOP-Except :

Видео

Исключения

Массивы указателей на функции

Исключения

Во время работы программы возможна такая ситуация, когда функция или какя-либо другая часть программы сталкивается со значениями, которые выходят за рамки допустимого применения этой программы. Это приводит либо к внезапной остановки программы, либо (что еще хуже) дальнейшия действия становятя непредсказуемыми.

Такие ситуации называются **исключительными** или **исключения.**

Исключительная стуация может возникнуть в какой-либо функции, которая сама по себе вызвана другой функцией и т.д. Однако решение как поступить в данной ситуации в процессе работы этой функции мы принять не можем. Скорее всего принять такое решение мы можем в той части программы, которая и осуществила вызов этой функции, возможно через вызовы других функций.

Возникает задача как передать информацию об исключительной ситуации в тот раздел программы где она может быть правильно обработана.

Для этого в языке C++ существует специальный оператор **throw**, который может сгенерить объект для передачи информации об исключении и передать в ту часть программы, которая непосредственно вызвала эту функцию.

Далее этой объект может быть **пойман** и обработан или передан на вызов выше.

Для того чтобы **поймать** объект-исключение, в C++ существует конструкция **try catch**. Объект, с которым передается информация об исключении может быть любой, лишь бы наша программа могла его правильно понять.

Простой пример.

```
#include <cstdlib>
#include <iostream>

using namespace std;
// Создадим какой-нибудь класс для
// передачи информации об исключительной ситуации
class MyException{

public:
// этот метод мы можем вызвать чтобы
// посмотреть что произошло
     virtual void trouble();
};
// не очень информативно
// зато понятно, что исключение случилось
void MyException::trouble(){
   cout<<"Что-то случилось\n";
};
// Функция собирается делить на 0!!
int part(int a, int b){
// проверим делитель
   if(b == 0)

// если он равен 0, сгенерим объект
// для передачи в вызывающую функцию
//  и передадим объект туда
   throw MyException();
// если произошла исключительная ситуация,
// функция дальше не выполняется и прерывается
// то есть, никакие значения не будут переданы

    return a/b;
}

int main(int argc, char *argv[])
{
  int k = 15, a , b;
   cin>>a>>b;
// попробуем исполнить part()
   try{
       k = part(a,b);
// если все нормально, k получит результат
// если будет исключение, оно может быть поймано
// мы лодны указать ТИП объекта-исключения, которое
// собираемся ловить
```


Поиск

Поиск

Раздел «Язык Си»

Главная
Зачем учить С?
Определения

Инструменты:

Поиск
Изменения
Index
Статистика

Разделы

Информация
Алгоритмы
Язык Си
Язык Ruby
Язык Ассемблера
El Judge
Парадигмы
Образование
Сети
Objective C

Logon>>

```
// В нашем случае это только MyException
    } catch(MyException pe){
// поймав объект, назовем его pe
// дальше можно обращаться к методам этого объекта
        pe.trouble();
          cout<<"Программа будет прервана\n";
// если программу здесь не перывать, и никаких
// других действий не делать, то
// программа продолжит выполняться дальше
     exit(1);
    }

    cout<<"k="<<k<<endl;
    return 0;
}
```

Пример более глубокого вызова, где возникает исключение

```
#include <cstdlib>
#include <iostream>

using namespace std;
// Создадим какой-нибудь класс для
// передачи информации об исключительной ситуации
class MyException{

public:
// этот метод мы можем вызвать чтобы
// посмотреть что произошло
     virtual void trouble();
};
// не очень информативно
// зато понятно, что исключение случилось
void MyException::trouble(){
   cout<<"Что-то случилось\n";
};
// Функция собирается делить на 0!!
int part(int a, int b){
// проверим делитель
   if(b == 0)

// если он равен 0, сгенерим объект
// для передачи в вызывающую функцию
//  и передадим объект туда
   throw MyException();
// если произошла исключительная ситуация,
// функция дальше не выполняется и прерывается
// то есть, никакие значения не будут переданы

    return a/b;
};
// здесь идет вызов part()
int  z(int a, int b){
   int tmp;
// пытаемся исполнить part()
   try{
    tmp = a + b + part(a,b);
   }catch(MyException ex){
//  z не знает что делать с 0
        cout<<"z передает исключение выше\n";

// передает объект MYException выше
        throw;
   }
};

int main(int argc, char *argv[])
{
  int k=15,a,b;
   cin>>a>>b;
   try{
      k=z(a,b);
    } catch(MyException pe){
        cout<<"main поймал исключение\n";
        pe.trouble();
   exit(1);
    }
```

Различные типы исключений. Как поймать все сразу

Для того, чтобы ситуация было понятнее можно определить разные классы объектов-исключения для обработки различных ситуаций.

Допустим у нас могут возникунть проблемы с делением на 0, нарушением диапазона и другие математические неприятности.

Создадим три класса для обработки таких исключений. Но заметим, что все они, в сущности, математические. Поэтому класс **MathEx** – будет базовый для всех.

Файл **try.h**

```
#include <cstdlib>
#include <iostream>
```

```
using namespace std;
// базовый класс всех наших исключений
class MathEx{
 public:
// виртуальная функция сообщений
        virtual void prerr(ostream&);
};
// класс для обозначения проблем с диапазоном
class DiaEx:public MathEx{
// сохраним проблемные значения
  int a,b;
public:
// конструктор нужен
    DiaEx(int f,int s);
// переопределяем виртуальную функцию
    void prerr(ostream&);
};
// класс для обозначения проблем с нулем
class ZeroEx:public DiaEx{
  public:
// тоже нужен инициализирующий конструктор
    ZeroEx(int f, int z);
//  так же переопределим функцию сообщений
   void prerr(ostream&);
};
```

Реализация

```
#include "try.h"

void MathEx::prerr(ostream& s){
    s<<"Math error\n";
};

ZeroEx::ZeroEx(int f, int z):DiaEx(f,z){
};

void ZeroEx::prerr(ostream& s){
   s<<"Zero problem, ";
    DiaEx::prerr(s);
    s<<'\n';
};

DiaEx::DiaEx(int f, int z){
   a = f;
   b = z;
};
void DiaEx::prerr(ostream& s){
        s<<"Diapazon! ";
   s<<a<<' '<<b<<endl;
};
```

Используем наши классы, чтобы поймать все исключения сразу

```
#include "try.h"
#include <math.h>
int part(int a, int b){
   if(b == 0)
      throw ZeroEx(a,b);
    return a/b;
};
int add(int a, int b){
   if( a < b)
     throw MathEx();
};

int sq(int a, int b){
   if (a < 0 && b >0  || b < 0 && a > 0)
      throw DiaEx(a,b);
    return sqrt(a*b);
};

int  z(int a, int b){
   int tmp;
   try{
    tmp=add(a,b) + sq(b,a) + part(a,b);
// ловить будем ССЫЛКУ
   }catch(MathEx& ex){
        cout<<"z передает исключение выше\n";
        throw;
   }
};

int main(int argc, char *argv[])
{
  int k=15,a,b;
   cin>>a>>b;
   try{
       k=z(a,b);
// Пойманая ССЫЛКА позволяет использовать
// виртуальную функцию prerr() ПОЛИМОРФНО
// то есть будет работать та функция,
// которая принадлежит конкретному исключению
    } catch(MathEx& pe){
```

```
        cout<<"main поймал исключение\n";
        pe.prerr(cout);
        exit(1);
    }

    cout<<"k="<<k<<endl;

}
```

Задачи

Задача 1.

Запустить и отладить пример. Проверить его на разных значениях a и b для получения различных исключительных ситуаций.

Исключения при создании объектов.

Рассмотрим пример, кода исходные данные не позволяют создать допустимый условиями задачи объект.

Треугольник на плоскости задан тремя точками. Мы должны рассмотреть все НЕВЫРОЖДЕННЫЕ треугольники. При создании таких объктов могут вызникать исключительные ситуации: все точки лежат на одной прямой, две или боьше точек совпадают и т.д.

Конструкторы объектов не возвращают никаких значений, поэтому корректно решить эту проблему помогут исключения.

```
#include "try.h"
#include <fstream>
#include <math.h>
// точка как структура
struct Point{
  int x, y;
  Point(int, int);
  Point();
};
// класс треугольников
class Triangle{
  Point pt[3]; // - вершины
  int len[3]; // длины сторон
public:
  Triangle(Point a[3]);
  void print();
};

Point::Point(int a, int b){
  x = a; y = b;
};
Point::Point(){
  x = y = 0;
};
// печать информации о треугольнике
void Triangle::print(){
  for (int i=0; i< 3; i++){
    cout<<'('<<pt[i].x<<","<<pt[i].y<<") ";
  }
  for (int i=0; i< 3; i++){
    cout<<len[i];
   if (i<2)
     cout<<", ";
  }
  cout<<endl;
};
// конструкто треугольника
Triangle::Triangle(Point a[3]){
   for( int i = 0; i < 3; i++){
   pt[i] = a[i];
     len[i] = (a[i].x - a[(i + 1)%3].x)*(a[i].x - a[(i + 1)%3].x)+(a[i].y - a[(i + 1)%3].y)*(a[i].y - a[(i + 1)%3].y);
    }
// проверка невырожденности треугольника
  for(int i = 0; i < 3; i++){
    if(sqrt(len[i])+sqrt(len[(i+1)%3]) <= sqrt(len[(i+2)%3]))
// если вырожден, то генерится исключение диапазона с парраметрами
      throw DiaEx(sqrt(len[i])+sqrt(len[(i+1)%3]),sqrt(len[(i+2)%3]));
  }
};

int main(){
  Point a1[3],b1[3];
  a1[0]=Point(0,0);
  a1[1]=Point(3,0);
  a1[2]=Point(0,4);
  b1[0]=Point(0,0);
  b1[0]=Point(-50,0);
  b1[0]=Point(100,0);
// попытка создать два треугольника из заданных точек
  try{
    Triangle a(a1);
    a.print();
    Triangle b((b1));
    b.print();
// первое исключение, которое сгенерится и будет поймано.
  }catch (MathEx& pe){
     pe.prerr(cout);
  }
```

```
}
```

Задача 2.

Для этой же задачи написать попытку создать динамические объекты треугольников. Обратиться к их методам после catch.

-- TatyanaOvsyannikova2011 - 04 Apr 2017