PEMROGRAMAN LANJUT

14 | Exception Handling

Team Teaching Pemrograman Lanjut

Fakultas Ilmu Komputer UB

Pendahuluan

- Kesalahan sering terjadi pada saat perancangan dan implementasi
- Kesalahan dikategorikan :
  - Syntak error menyebabkan kesalahan kompilasi
  - Semantic error , program menghasilkan keluaran yang tidak sesuai dengan harapan
  - Run-time error, kebanyakan mengakibatkan terminasi program secara tidak normal atau bahkan sistem crash. Misal : penggunaan tipe data yang salah.

Error Handling

- Setiap program yang berada dalam suatu kondisi yang tidak normal – Error Conditions.
- Program yang ‘baik’ harus dapat menangani kondisi ini.
- C++ menyediakan suatu mekanisme untuk menangani kondisi ini - *exceptions*

Exception

- Exception merupakan suatu keadaan yang disebabkan oleh runtime error **dalam** program.
- Memungkinkan **kesalahan** ditangani tanpa harus ‘mengotori’ program (dengan rutin yang menangani **kesalahan**)
- Memungkinkan pemisahan **penanganan kesalahan** dengan program utama

Apa yang terjadi jika terjadi kesalahan?

- Secara otomatis akan dilempar sebuah object yang disebut dgn ***exception***.
- Exception dapat diproses lebih lanjut oleh fungsi-fungsi yang siap menangani kesalahan.
- Proses pelemparan exception disebut dgn ***throwing exception***.
- Proses penerimaan exception disebut dengan ***catch exception***.

Contoh Exception:

- File yang akan dibuka tidak ada.
- Koneksi jaringan putus.
- Pembagian bilangan dengan 0
- Pengisian elemen array diluar ukuran array
- Kegagalan koneksi database
- Operan yg akan dimanipulasi out of prescribed range

Keywords

- try
- catch
- throw
- throws

Bentuk penggunaan

```
try{
   /* kode yang memungkinkan terjadinya eksepsi
}
catch(TipeEksepsi1 &obyekEksepsi)1{
   /* kode untuk menangani eksepsi1
}
//…
catch(TipeEksepsiN &obyekEksepsiN){
    /* kode untuk menangani eksepsiN
}
```

Try & Catch?

- Blok **try** : digunakan untuk menempatkan kode-kode program yang mengandung kode program yang mungkin melemparkan exception.
- Blok **catch** : digunakan untuk menempatkan kode-kode program yang digunakan untuk menangani sebuah exception tertentu.

Tanpa Exception

```
#include<iostream>
using namespace std;
double quotient(int numerator,int denominator){
      return static_cast<double>(numerator)/denominator;
}
int main(){
      int number1=4;
      int number2=0;
      double result;
      result=quotient(number1,number2);
      cout<<"the quotient is "<<result<<endl;
}
```

DivideByZeroException.h

```
#include<iostream>
#include<stdexcept>
using namespace std;

class DivideByZeroException:public runtime_error
{
     public:

     DivideByZeroException():runtime_error("attemped to
divide by zero"){}

};
```

Main.cpp

```
#include<iostream>
using namespace std;
#include "DivideByZeroException.h"
double quotient(int numerator,int denominator){
      if(denominator==0)
      throw DivideByZeroException();
      return static_cast<double>(numerator)/denominator;
}
```

Main.cpp (Lanjutan)

```
int main(){
	int number1=4;
	int number2=0;
	double result;
	try{
		result=quotient(number1,number2);
		cout<<"the quotient is "<<result<<endl;
	}
	catch(DivideByZeroException &divideByZeroException){
		cout<<"Exception
occured:"<<divideByZeroException.what()<<endl;
	}
}
```