

Lecture 9

Exception

- ❑ An exception is a unusual, often unpredictable event, detectable by software or hardware, that requires special processing occurring at runtime.
- ❑ In C++, a variable or class object that represents an exceptional event

Handling Exception

- Without handling,
 - Program crashes
 - Falls into unknown state
- An **exception handler** is a section of program code that is designed to execute when a particular exception occurs
 - Resolve the exception
 - Lead to known state, such as exiting the program

Standard Exceptions

- ❑ Exceptions Thrown by the Language
 - *new*
- ❑ Exceptions Thrown by Standard Library Routines
- ❑ Exceptions Thrown by user code, using *throw* statement

The *throw* Statement

- **Throw**: to signal the fact that an exception has occurred; also called **raise**

ThrowStatement

throw Expression

The *try-catch* Statement

- How one part of the program catches and processes the exception that another part of the program throws.

TryCatchStatement

```
try
  Block
catch (FormalParameter)
  Block
catch (FormalParameter)
```

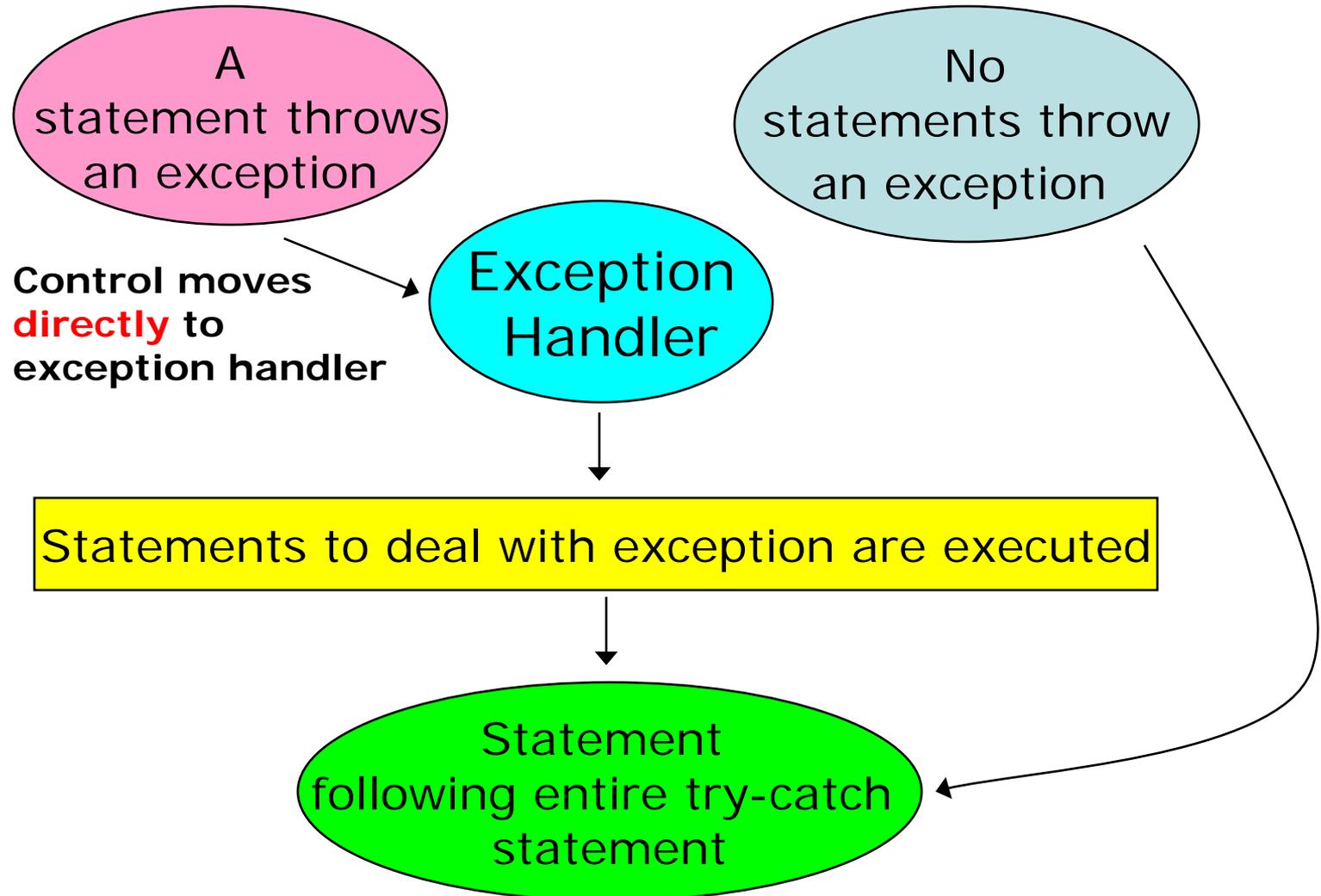
FormalParameter

```
{  DataType VariableName
  ...
```

Example of a try-catch Statement

```
try
{
    // Statements that process personnel data and may
    // throw exceptions of type int, string, and SalaryError
}
catch ( int )
{
    // Statements to handle an int exception
}
catch ( string s )
{
    cout << s << endl; // Prints "Invalid customer age"
    // More statements to handle an age error
}
catch ( SalaryError )
{
    // Statements to handle a salary error
}
```

Execution of *try-catch*



Throwing an Exception to be Caught by the Calling Code

```
void Func3()
```

```
{
```

```
  try
```

```
  {
```

```
    Func4();
```

```
  }
```

```
  catch ( ErrType )
```

```
  {
```

```
  }
```

```
}
```

**Function
call**

**Normal
return**

**Return from
thrown
exception**

```
void Func4()
```

```
{
```

```
  if ( error )
```

```
    throw ErrType();
```

```
}
```

Practice: Dividing by ZERO

Apply what you know:

```
int Quotient(int numer,      // The numerator
              int denom ) // The denominator
{
    if (denom != 0)
        return numer / denom;
    else
        // What to do?? do sth. to avoid program          //crash
}
```

A Solution

```
int Quotient(int numer, // The numerator
             int denom ) // The denominator
{
    if (denom == 0)
        throw DivByZero();
        //throw exception of class DivByZero
    return numer / denom;
}
```

A Solution

```
// quotient.cpp -- Quotient
program
#include<iostream.h>
#include <string.h>
int Quotient( int, int );
// Exception class
class DivByZero {};
int main()
{
    int numer; // Numerator
    int denom; // Denominator

    //read in numerator
    and denominator
```

```
while(cin)
{
    try
    {
        cout << "Their quotient: "
            <<Quotient(numer,denom)<<endl;
    }
    //exception handler
    catch ( DivByZero)
    {
        cout<<"Denominator can't be 0"<<
endl;
    }
    // read in numerator and denominator
}
return 0;
}
```

Take Home Message

- An exception is a unusual, often unpredictable event that requires special processing occurring at runtime
 - Throw
 - Try-catch