在C#中，使用类表示异常。所有的异常类都由内置的异常类Exception派生而来。

C#异常处理的4个关键字：try、catch、throw和finally。

try-catch必须结合使用

例：程序exception\_test1

// Copyright 2016.刘珅珅

// author：刘珅珅

// 异常处理

using *System*;

using *System*.*Collections*.*Generic*;

using *System*.*Linq*;

using *System*.*Text*;

using *System*.*Threading*.*Tasks*;

namespace exception\_test1

{

class ExceptionTest

{

static void Main(string[] args)

{

int[] nums = new int[5];

try

{

*Console*.*WriteLine*("Before exception is generated.");

for (int i = 0; i < 10; i ++)

{

nums[i] = i;

*Console*.*WriteLine*("nums[{0}] : {1}", i, nums[i]);

}

*Console*.*WriteLine*("This won't be displayed.");

}

catch (*IndexOutOfRangeException*)

{

*Console*.*WriteLine*("Index out-out-bound.");

}

}

}

}

输出结果为：

Before exception is generated.

nums[0] : 0

nums[1] : 1

nums[2] : 2

nums[3] : 3

nums[4] : 4

Index out-out-bound.

多个catch的情况：

例：程序exception\_test2

// Copyright 2016.刘珅珅

// author：刘珅珅

// 异常处理

using *System*;

using *System*.*Collections*.*Generic*;

using *System*.*Linq*;

using *System*.*Text*;

using *System*.*Threading*.*Tasks*;

namespace exception\_test2

{

class ExceptionTest

{

static void Main(string[] args)

{

int[] nums = { 4, 8, 16, 32, 64, 128, 256, 512};

int[] denom = { 2, 0, 4, 4, 0, 8 };

for (int i = 0; i < nums.*Length*; ++i)

{

try

{

*Console*.*WriteLine*(nums[i] + " / "

+ denom[i] + " is "

+ nums[i] / denom[i]);

}

catch (*DivideByZeroException*)

{

*Console*.*WriteLine*("Can't divide by Zero!");

}

catch (*IndexOutOfRangeException*)

{

*Console*.*WriteLine*("No matching element found.");

}

}

}

}

}

输出结果：

4 / 2 is 2

Can't divide by Zero!

16 / 4 is 4

32 / 4 is 8

Can't divide by Zero!

128 / 8 is 16

No matching element found.

No matching element found.

从输出结果可以看出，使用try-catch，当抛出异常时，程序并不会终止。

抛出异常：throw

finally语句中定义的代码，无论try-catch块如何结束，都会被执行，即使try-catch中包含return语句。

例：程序exception\_test3

// Copyright 2016.刘珅珅

// author：刘珅珅

// 异常处理try-catch-finally

using *System*;

using *System*.*Collections*.*Generic*;

using *System*.*Linq*;

using *System*.*Text*;

using *System*.*Threading*.*Tasks*;

namespace exception\_test3

{

class UseFinally

{

public static void GenException(int what)

{

int t;

int[] nums = new int[2];

*Console*.*WriteLine*("Receiving " + what);

try

{

switch (what)

{

case 0:

t = 10;

break;

case 1:

nums[4] = 4;

break;

case 2:

return;

}

}

catch (*DivideByZeroException*)

{

*Console*.*WriteLine*("Can't divide by Zero!");

return;

}

catch (*IndexOutOfRangeException*)

{

*Console*.*WriteLine*("No matching element found.");

}

finally

{

*Console*.*WriteLine*("Leaving try.");

}

}

}

class ExceptionTest

{

static void Main(string[] args)

{

for (int i = 0; i < 3; ++i)

{

UseFinally.GenException(i);

*Console*.*WriteLine*();

}

}

}

}

输出结果：

Receiving 0

Leaving try.

Receiving 1

No matching element found.

Leaving try.

Receiving 2

Leaving try.

从结果看出：无论在try-catch如何结束，即使使用了return语句，仍然会执行finally中的语句。

派生异常类：

例：程序exception\_test4

// Copyright 2016.刘珅珅

// author：刘珅珅

// 派生异常类

using *System*;

using *System*.*Collections*.*Generic*;

using *System*.*Linq*;

using *System*.*Text*;

using *System*.*Threading*.*Tasks*;

namespace exception\_test4

{

class ExceptionTest

{

class RangeArrayException : *Exception*

{

public RangeArrayException() : base() { }

public RangeArrayException(string message) : base(message) { }

public RangeArrayException(string message, *Exception* inner\_exception)

: base(message, inner\_exception) { }

public RangeArrayException(*System*.*Runtime*.*Serialization*.*SerializationInfo* info,

*System*.*Runtime*.*Serialization*.*StreamingContext* context)

: base(info, context) { }

public override string ToString()

{

return *Message*;

}

}

class RangeArray

{

int[] array;

int lower\_bound;

int upper\_bound;

public int Length { get; private set; }

public RangeArray(int low, int high)

{

++high;

if (high <= low)

{

throw new RangeArrayException("Low index not less than high.");

}

array = new int[high - low];

Length = high - low;

lower\_bound = low;

upper\_bound = --high;

}

public int this[int index]

{

get

{

if (Ok(index))

{

return array[index - lower\_bound];

}

else

{

throw new RangeArrayException("Range Error.");

}

}

set

{

if (Ok(index))

{

array[index - lower\_bound] = value;

}

else

{

throw new RangeArrayException("Range Error.");

}

}

}

private bool Ok(int index)

{

if (index >= lower\_bound && index <= upper\_bound)

{

return true;

}

return false;

}

}

static void Main(string[] args)

{

try

{

RangeArray obj1 = new RangeArray(-2, 2);

for (int i = -2; i < 10; ++i)

{

*Console*.*WriteLine*(obj1[i]);

}

}

catch (RangeArrayException ex)

{

*Console*.*WriteLine*(ex);

}

}

}

}

输出结果为：

0

0

0

0

0

Range Error.

try-catch语句在捕获异常时，有一定的顺序要求。

例：程序exception\_test5

// Copyright 2016.刘珅珅

// author：刘珅珅

// 捕获异常的顺序

using *System*;

using *System*.*Collections*.*Generic*;

using *System*.*Linq*;

using *System*.*Text*;

using *System*.*Threading*.*Tasks*;

namespace exception\_test5

{

class ExceptA : *Exception*

{

public ExceptA(string message) : base(message) { }

public override string ToString()

{

return *Message*;

}

}

class ExceptB : ExceptA

{

public ExceptB(string message) : base(message) { }

public override string ToString()

{

return *Message*;

}

}

class ExceptionTest

{

static void Main(string[] args)

{

for (int i = 0; i < 3; ++i)

{

try

{

if (i == 0)

{

throw new ExceptA("Caught and ExceptA exception.");

}

else if (i == 1)

{

throw new ExceptB("Caught an ExceptB exception.");

}

else

{

throw new *Exception*();

}

}

catch (ExceptB ex)

{

*Console*.*WriteLine*(ex);

}

catch (ExceptA ex)

{

*Console*.*WriteLine*(ex);

}

catch (*Exception* ex)

{

*Console*.*WriteLine*(ex);

}

}

}

}

}

在程序里，ExceptB继承自ExceptA，ExceptA继承自Exception。ExceptB的catch语句必须位于ExceptA的catch语句之前，Exception这个捕获所有异常的catch语句必须位于最后面。

checked和unchecked关键字

C#可以使用checked和unchecked关键字由用户指定程序在进行算术运算溢出是否产生异常。

checked:产生溢出则抛出OverflowException

unchecked:忽略溢出

checked使用：

checked(expr) // 检查表达式

checked { // 检测语句块

}

unchecked使用：

unchecked(expr) // 检查表达式

unchecked { // 检测语句块

}

例：程序exception\_test6

// Copyright 2016.刘珅珅

// author：刘珅珅

// checked和unchecked

using *System*;

using *System*.*Collections*.*Generic*;

using *System*.*Linq*;

using *System*.*Text*;

using *System*.*Threading*.*Tasks*;

namespace exception\_test6

{

class ExceptionTest

{

static void Main(string[] args)

{

byte a = 127;

byte b = 127;

byte result;

try

{

result = unchecked((byte)(a \* b));

*Console*.*WriteLine*("Unchecked result: " + result);

result = checked((byte)(a \* b));

*Console*.*WriteLine*("Checked result: " + result);

}

catch (*OverflowException* ex)

{

*Console*.*WriteLine*(ex);

}

}

}

}

输出结果为:

Unchecked result: 1

System.OverflowException: 算术运算导致溢出。

在 exception\_test6.ExceptionTest.Main(String[] args) 位置 c:\study\CSharp-stu

dy\exception\_test6\ExceptionTest.cs:行号 25

从结果可以看出，使用unchecked运算符时，会忽略异常，使用checked运算符会抛出异常。