可空类型：

可空类型是一个特殊的值类型，由结构表示。

可空类型是System.Nullable<T>类型的对象，其中T必须是一个非可空的值类型。

可空类型的取值范围和特征与其底层类型相同，只是它可以用来指示变量不包含任何值。

例：程序unsafe\_test7

// Copyright 2016.刘珅珅

// author：刘珅珅

// 可空类型

using *System*;

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

using *System*.*Linq*;

using *System*.*Text*;

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

namespace unsafe\_test7

{

class NullableTest

{

static void Main(string[] args)

{

// 两种初始化方式相同

*System*.*Nullable*<int> count = null;

int? count1 = null;

if (count.*HasValue*)

{

*Console*.*WriteLine*("count has this value: " + count.*Value*);

}

else

{

*Console*.*WriteLine*("count has no value.");

}

count = 100;

if (count.*HasValue*)

{

*Console*.*WriteLine*("count has this value: " + count.*Value*);

}

else

{

*Console*.*WriteLine*("count has no value.");

}

}

}

}

输出为：

count has no value.

count has this value: 100

可空对象可以用于表达式中，混用可空类型和非可空类型结果是一个可空类型。

例：程序unsafe\_test8

// Copyright 2016.刘珅珅

// author：刘珅珅

// 表达式中的可空类型

using *System*;

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

using *System*.*Linq*;

using *System*.*Text*;

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

namespace unsafe\_test8

{

class NullableTest

{

static void Main(string[] args)

{

int? count = null;

int? result = null;

int n = 10; // 非可空类型

result = count + n; // 可空类型和非可空类型的结果为可空类型

if (result.*HasValue*)

{

*Console*.*WriteLine*("result has value: " + result.*Value*);

}

else

*Console*.*WriteLine*("result has no value");

count = 100;

result = count + n;

if (result.*HasValue*)

{

*Console*.*WriteLine*("result has value: " + result.*Value*);

}

else

*Console*.*WriteLine*("result has no value");

// 可空对象赋值给非可空对象

double? balance = null;

double current\_balance; // 非可空对象

// 如果balance为空，则赋值0.0给current\_balance

current\_balance = balance ?? 0.0;

}

}

}

输出结果为：

result has no value

result has value: 110

可空对象使用关系和逻辑运算符

可空类型用在关系表达式中，与非可空类型的用法一样，可空类型可以和非可空类型混用。但有一个附加的规则，在使用关系运算符“<”、“>”、或“<=”或“>=”比较两个可空对象时，只要其中一个对象的值为null，则结果为false。

例：程序unsafe\_test9

// Copyright 2016.刘珅珅

// author：刘珅珅

// 可空对象使用关系和逻辑运算符

using *System*;

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

using *System*.*Linq*;

using *System*.*Text*;

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

namespace unsafe\_test9

{

class NullableTest

{

static void Main(string[] args)

{

int? count = null;

int? length = 16;

*Console*.*WriteLine*(count < length); // False

*Console*.*WriteLine*(count == length); // False

*Console*.*WriteLine*(count > length); // False

int k = 8;

*Console*.*WriteLine*(count < k); // False

*Console*.*WriteLine*(count > k); // False

*Console*.*WriteLine*(k < length); // True

}

}

}

从结果可以看出，只要其中一个可空类型为null，则无论比较的另一个对象是可空还是非可空，无法是否有值，表达式返回的结果均为false。

然而，在使用“==”或“!=”运算符可以测试可空对象是否包含null值。

if (count == null) // 表达式为True

对于bool?可空类型

例：程序unsafe\_test9

bool? p = true;

bool? q = null;

*Console*.*WriteLine*(p | q); // True

*Console*.*WriteLine*(p & q); // null

p = false;

*Console*.*WriteLine*(p | q); // null

*Console*.*WriteLine*(p & q); // False

p = null;

*Console*.*WriteLine*(p | q); // null

*Console*.*WriteLine*(p & q); // null

*Console*.*WriteLine*(!p); // null