Subject: Dumb bug. Improper use of Null
Posted by koldo on Thu, 05 Sep 2019 10:13:52 GMT
View Forum Message <> Reply to Message

```
int a = Null;
int64 b = a;
if (IsNull(b))
 Cout() << "I wanted this";
else
 Cout() << "Oh no!";
```

Subject: Re: Improper use of Null
Posted by Sender Ghost on Mon, 09 Sep 2019 18:23:03 GMT
View Forum Message <> Reply to Message

Hello Iñaki.

I think, there is some explanation in "U++ Core Tutorial" about int and int64 types, where Null "defined as lowest number the type can represent".

Following example may show how it works:

```
#include <Core/Core.h>
#include <iostream>

using namespace Upp;

CONSOLE_APP_MAIN
{
 int i = Null;
 int64 i64 = Null;
 double d = Null;
 Value v = Null;

 std::cout << "i = " << i << std::endl
  << "i64 = " << i64 << std::endl
  << "d = " << d << std::endl
  << "v = " << ~AsString(v) << std::endl;

 if ((i == i64) && (i64 == d) && (d == v))
  NEVER();
 else
  Cout() << "This is how it works\n";

 Value vi = i,
  vi64 = i64,
  vd = d,
```

```
  vv = v;

 if ((vi == vi64) && (vi64 == vd) && (vd == vv))
  Cout() << "This is how it works\n";
 else
  NEVER();
}
```

With following results:
i = -2147483648
i64 = -9223372036854775808
d = -1e+308
v =
This is how it works
This is how it works
where i is assigned to INT_NULL (equal to INT_MIN) and i64 is assigned to INT64_NULL (equal to INT64_MIN) values. In other words, i > i64 and int64 type may include INT_NULL (which is not equal to INT64_NULL) in its range.

I guess, possible to use Value type for intermediate Null value, e.g. to "transfer" Null value from some type to another type:

```
#include <Core/Core.h>

using namespace Upp;

CONSOLE_APP_MAIN
{
 int a = Null;
 int64 b = Value(a);

 if (IsNull(b))
  Cout() << "I wanted this\n";
 else
  Cout() << "Oh no!\n";
}
```

Other examples

```
#include <Core/Core.h>

using namespace Upp;

#define PRINT_RESULT(x) \
 if (IsNull(x)) \
  Cout() << "I wanted this\n"; \
 else \
  Cout() << "Oh no!\n";
```

```
#define TEST1(type) { \
 Cout() << typeid(type).name() << ": "; \
 int a = Null; \
 type b = a; \
 PRINT_RESULT(b); \
}

#define TEST2(type) { \
 Cout() << typeid(type).name() << ": "; \
 int a = Null; \
 type b = IsNull(a) ? Null : a; \
 PRINT_RESULT(b); \
}

#define TEST3(type) { \
 Cout() << typeid(type).name() << ": "; \
 int a = Null; \
 type b; \
 if (IsNull(a)) \
  b = Null; \
 else \
  b = a; \
 PRINT_RESULT(b); \
}

#define TEST4(type) { \
 Cout() << typeid(type).name() << ": "; \
 int a = Null; \
 type b = Value(a); \
 PRINT_RESULT(b); \
}

#define TYPES(d) Cout() << #d << '\n'; \
 d(int); \
 d(int64); \
 d(double); \
 d(Value);

CONSOLE_APP_MAIN
{
 TYPES(TEST1);
 TYPES(TEST2);
 TYPES(TEST3);
 TYPES(TEST4);
}
```

Results:

```
TEST1
i: I wanted this
x: Oh no!
d: Oh no!
N3Upp5ValueE: I wanted this
TEST2
i: I wanted this
x: Oh no!
d: Oh no!
N3Upp5ValueE: I wanted this
TEST3
i: I wanted this
x: I wanted this
d: I wanted this
N3Upp5ValueE: I wanted this
TEST4
i: I wanted this
x: I wanted this
d: I wanted this
N3Upp5ValueE: I wanted this
```

```
#include <Core/Core.h>

using namespace Upp;

CONSOLE_APP_MAIN
{
 int a = Null;
#if 1
 int64 b = a; // Not Null [*]
#elif 0
 double b = a; // Not Null [*]
#elif 0
 Value b = a; // Null [*]
#elif 0
 int64 b = Null; // Null
#elif 0
 double b = Null; // Null
#elif 0
 Value b = Null; // Null
#elif 0
 int64 b = IsNull(a) ? Null : a; // Not Null [*]
#elif 0
 double b = IsNull(a) ? Null : a; // Not Null [*]
#elif 0
 Value b = IsNull(a) ? Null : a; // Null [*]
```

```
#elif 0
 int64 b;
 if (IsNull(a))
  b = Null; // Null
 else
  b = a;
#elif 0
 double b;
 if (IsNull(a))
  b = Null; // Null
 else
  b = a;
#elif 0
 Value b;
 if (IsNull(a))
  b = Null; // Null
 else
  b = a;
#elif 0
 int64 b = IsNull(a) ? INT64_NULL : a; // Null
#else
 double b = IsNull(a) ? DOUBLE_NULL : a; // Null
#endif

 if (IsNull(b))
  Cout() << "I wanted this\n";
 else
  Cout() << "Oh no!\n";
}
```

Subject: Re: Dumb bug. Improper use of Null
Posted by mirek on Mon, 09 Sep 2019 18:59:27 GMT
View Forum Message <> Reply to Message

Yeah, that is unfortunate drawback of allowing fundamental types to be Null... There unfortunately is no way how to make this work.

That said, perhaps we could introduce something like

int64 x = NvlTo<int64>(s);

Conversion to Value works as well, but is a bit slow.

Mirek

---

mirek wrote on Mon, 09 September 2019 18:59That said, perhaps we could introduce something like

int64 x = NvlTo<int64>(s);
Something like this?

```
#include <Core/Core.h>

using namespace Upp;

template <class T, class C>
T NvlTo(const C& x)
{
 if (IsNull(x))
  return Null;
 return x;
};

template <class T>
void Print(const T& x)
{
 if (IsNull(x))
  Cout() << "I wanted this\n";
 else
  Cout() << x << '\n';
}

CONSOLE_APP_MAIN
{
#if 1
 int a = Null;
 int64 b = NvlTo<int64>(a);
 Print(b);

 a = -10;
 b = NvlTo<int64>(a);
 Print(b);
#else
 int a = Null;
 int64 b, c = 0;
 const int n = 1000000000;
 {
  RTIMING("NvlTo");
  for (int i = 0; i < n; ++i) {
   b = NvlTo<int64>(a);
```

```
   c += b + 1;
  }
 }
 Print(c);
 ASSERT(c == n);

 c = 0;
 {
  RTIMING("Value");
  for (int i = 0; i < n; ++i) {
   b = Value(a);
   c += b + 1;
  }
 }
 Print(c);
 ASSERT(c == n);
#endif
}
```

With following results:

I wanted this
-10

Thanks.

Subject: Re: Improper use of Null
Posted by koldo on Wed, 30 Oct 2019 07:11:44 GMT
View Forum Message <> Reply to Message

"A ship in the beach is a lighthouse to the sea" :)

Other scenario to watch out for:

```
MyFunction(double val) {
 if (IsNull(val))
  Cout() << "Null";
 else
  Cout() << "Not Null";
}
...
MyFunction(Null);  // "Null"
MyFunction(false ? 1 : Null); // "Not Null". The call sets INT_NULL instead of DOUBLE_NULL
```
