From Clever code to Better Code

Dror Helper

Senior Microsoft Specialist Architect
AWS

A long time ago, in an office no that far from here...

It has started just like any other morning...

A software developer having drank his mid-morning coffee, opens his trusty IDE, when suddenly he comes across a strange sequence of symbols...

```
#define GET_VAL(val, type) \
{ \
    ASSERT(( pIP + sizeof(type)) <= pMethodEnd); \
    val = ( *((type *&)(pIP))++); \
}
```

Home

PUBLIC

Stack Overflow

Tags

Users

Jobs

What is the most hard to understand piece of C++ code you know? [closed]

Ask Question

asked 10 years, 7 months ago

viewed 29,889 times

active 3 months ago

19

17

Today at work we came across the following code (some of you might recognize it):

```
#define GET_VAL( val, type ) \
    {                                                \
        ASSERT( ( pIP + sizeof(type) ) <= pMethodEnd ); \
        val = ( *((type *&)(pIP))++ );               \
    }
```

Basically we have a byte array and a pointer. The macro returns a reference to a variable of type and advance the pointer to the end of that variable.

It reminded me of the several times that I needed to "think like a parser" in order to understand C++ code.

Do you know of other code examples that caused you to stop and read it several times till you managed to grasp what it was suppose to do?

c++

share improve this question

edited May 3 '15 at 2:45

community wiki
5 revs, 3 users 100%
Dror Helper

Featured on Meta

Unicorn Meta Zoo #3: How do we grade questions?

Should we burninate the [eject] tag?

Related

2945 What are the differences between a pointer variable and a reference variable in C++?

862 What are POD types in C++?

872 What are the rules about using an underscore in a C++ identifier?

788 What are C++ functors and their uses?

1640 How can I profile C++ code running on Linux?

1397 What is the effect of extern “C” in C++?

Why we write clever code?

Because we can

Because we must

We don't know any better

Performance

```
                                                          12
void send(short* from, short* to, size_t count) {
    int n = (count + 7) / 8; n = 2
    switch (count % 8) {<-- 4
--->case 0: do { *to = *from++;
--->case 7:      *to = *from++;
--->case 6:      *to = *from++;
--->case 5:      *to = *from++;
--->case 4:      *to = *from++;
--->case 3:      *to = *from++;
--->case 2:      *to = *from++;
--->case 1:      *to = *from++;
            } while (--n > 0); n = 1
    }
}
```

Reality check

We write clever code because it makes us feel good

```
/* Most developers love being clever */
```

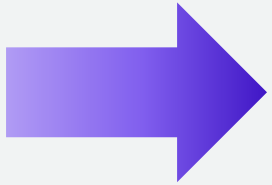
There is more than one way to write Clever code

```
public void int GetNextSize(int i)
{
    return i > 0 ? i << 2 : ~(i << 2) + 1;
}
```

```
public void int GetNextSize(int i)
{
    return Math.Abs(i * 4);
}
```

```
if((i & (i - 1) == 0)
{
   ...
}
```

```
1000 & 0111 --> 0
```



```
if(IsPowerOfTwo(i))
{
   ...
}

public bool IsPowerOfTwo(int i)
{
    return ((i & (i - 1) == 0;
}
```

Low level programming

- ✓ Avoid when you can
- ✓ Refactor to show intent

Compressed code

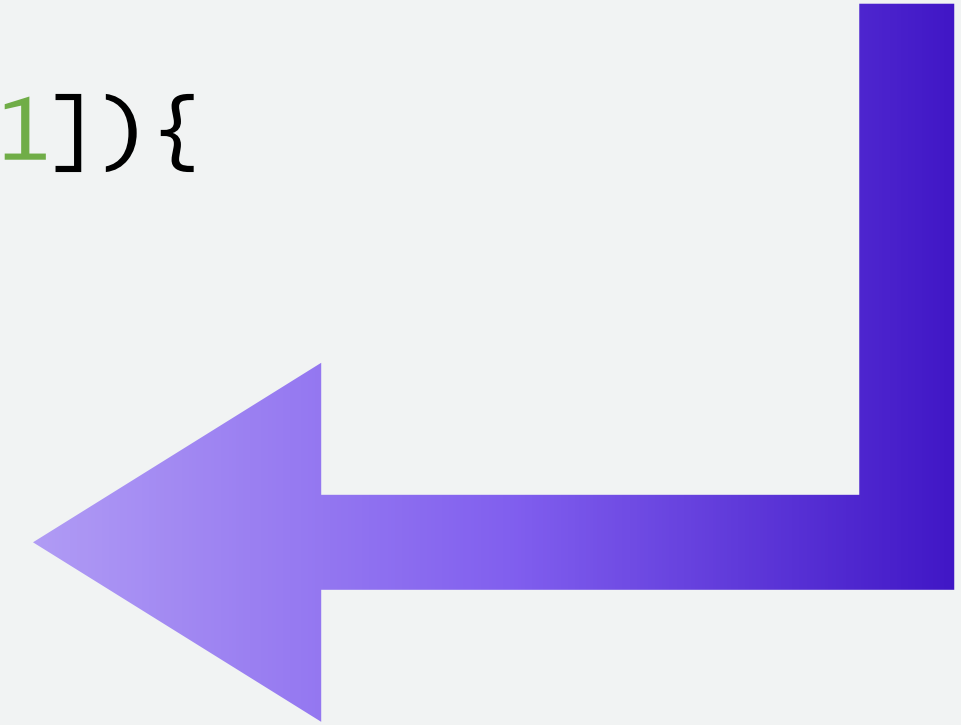
```
a = y[0]>y[1] ? (b=1,0) : (b=0,1);

a = (y[0] > y[1] ?
     (Func<int>)(() => { b = 1; return 0; }) :
                 () => { b = 0; return 1; }))
     .Invoke();
```

Uncompress code

```
a = y[0]>y[1] ? (b=1,0) : (b=0,1);

if(y[0] > y[1]){
    a = 0;
    b = 1;
} else {
    a = 1;
    b = 0;
}
```



Abusing the trinary operator

```
var name = ((GetAllNamesAsDelimitedString().contains(name) ?
                name: (GetEntityOperationMap().Count() == 0) ?
                null : GetEntityOperationMap().get(name))));
```

“Don’t try to write everything on the same line”

Tricky loops 

```
for (Dog dog = animal as Dog; dog != null; dog = null) {
    dog.bark();
}
```

```
Dog dog = animal as Dog;
if(dog != null)
{
    dog.Bark();
}
```

aws

Tricky loops 

```
int i;

for(i = 0 ; callMe() != null ; i++);
```

```
for(n = 0 ; n != 0;) {
     n = ReadNewValue();
     // do something
}
```

```
do
{
     n = ReadNewValue();
     // do something
} while(n != 0);
```

```
do{
      // code goes here
      if(some condition)
          break;
      // ...
       if(other condition)
          break;
      // ...
} while(false)
// cleanup here
```

```
try{
      // code goes here
      if(some condition)
            return;
      // ...
       if(other condition)
            return;
      // ...
} finally {
// cleanup here
```

```
public class Tree<T> {
    public T Value { get; }
    public Tree<T> Left { get; }
    public Tree<T> Right { get; }

    public void Find(T value)
    {
        if(Value.Equals(value))
            throw new FoundElementException<T>(this);

        Left?.Find(value);
        Right?.Find(value);
    }
}
```

Creative (ab)use

Never surprise code readers!

Think of a better, expected way to express yourself

```
public Task<Client> GetClientAsync(string clientId) {
    var client = _clientDal.FindClientById(clientId);

    if(client == null) {
        var client = CreateNewClient();

        await _clientDal.SaveClient(client);
        _notificationQueue.Push(NewClientCreatedNotification);
        ...
        var company = new Company(clientId, "temp");
        var account = CreateAccount(clientId, company);
        _accountDao.Save(account);
        _operationEvents.Send(new Notification(company, account));
    }

    return client;
}
```

Complex != Clever

Follow best practices/design patterns

Use clean code guidelines

And your common sense

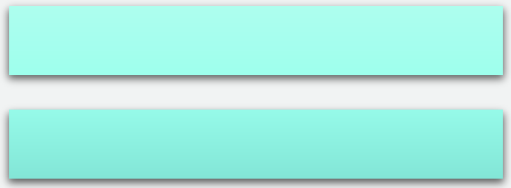
```
/* Otherwise: you'll regret it.
Maybe not today. Maybe not tomorrow,
but soon and for the rest of your life. */
```

```
class NotMyClass
{
    private string privateField;
}

class MyClass
{
    public string publicField;
}
```

```
[StructLayout(LayoutKind.Explicit)]
struct FastReflectionAdapter
{
    [FieldOffset(0)]
    internal object o1;

    [FieldOffset(0)]
    internal MyClass o2;
}
```



```
var adapter = new FastReflectionAdapter { o1 = new NotMyClass("Foo") };
var privateValue = adapter.o2.publicField;
```

aws

https://twitter.com/korifey_ad/status/1169210959426183168

What's so wrong with a little "cleverness"?

Writing < Reading < Debugging

"A clever person solves a problem.
A wise person avoids it."

Albert Einstein

What about performance?

```
void calculate(struct salesinfo* sales)
{
    jmp_buf buffer;
    int i=setjmp(buffer);

    if (!(i<sales->count))
        RETURN_NOTHING;

    addToSubtotal(sales->values[i]);

    if (i<sales->count)
    {
        longjmp(buffer,i+1);
    }
}
```

```
void calculate(struct salesinfo*sales)
{
    for(int i = 0; i < sales->count; i++)
    {
        addToSubtotal(sales->values[i]);
    }
}
```

Real optimizations

```
float InvSqrt(float x)
{
    float xhalf = 0.5f * x;
    int i = *(int*)&x;
    i = 0x5f375a86 - (i >> 1);
    x = *(float*)&i;
    x = x * (1.5f - xhalf * x * x);

    return x;
}
```

Is clever code good or evil?

```
OnPropertyChanged(GetPropertyName<User>(u => u.Address))
```

```
public string GetPropertyName<T>(Expression<Func<T, object>> property)
{
    LambdaExpression lambda = (LambdaExpression)property;
    MemberExpression memberExpression;

    if (lambda.Body is UnaryExpression)
    {
        UnaryExpression unaryExpression = (UnaryExpression)(lambda.Body);
        memberExpression = (MemberExpression)(unaryExpression.Operand);
    }
    else
    {
        memberExpression = (MemberExpression)(lambda.Body);
    }

    return ((PropertyInfo)memberExpression.Member).Name;
}
```

```
void parseCode(BYTE* pIp, BYTE* pMethodEnd)
{
    int valueType = getValueType(pIp)

    while(pIp < pMethodEnd)
    {
        switch(valueType){
            case ValueType::INT:
                int value;
                GET_VAL(value, int)
                // do something with value
            break;
            case ValueType::BYTE:
                    ...
        }
    }
}
```

Thank you!

Dror Helper
dhelper@amazon.com
@dhelper