

C++ Compiler behavior regarding struct constructors

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.vc/2004-12/0388.html>

From: Karl M (*Karl.M_at_email.com*)

Date: 12/19/04

Date: Sun, 19 Dec 2004 08:09:55 -0500

Hi everyone,

I just notice some strange behaviors on the MS C++ compiler regarding struct default constructor, see the example bellow:

```
struct MyStruct
```

```
{  
    int a[5];  
};
```

```
class MyClass
```

```
{  
public:  
    MyClass();  
    ~MyClass();
```

```
protected:
```

```
    MyStruct m_MyStruct;  
};
```

```
///  
//#1:
```

```
MyClass::MyClass() //The default ctor for the class does not init the struct  
with NULL(maybe there is no default ctor for struct yet?)
```

```
{  
}
```

```
///  
//#2 if I define the class ctor like this:
```

```
MyClass::MyClass() : m_MyStruct() //The default ctor for the class does init  
the struct with NULL(call m_MyStruct() default ctor, now there is one!)
```

```
{  
}
```

```
///  
//#3 or like this:
```

```
MyClass::MyClass()
```

```
{  
    m_MyStruct = MyStruct(); // dose init the struct with NULL using a local  
stack var init with NULL  
}
```

```
///  
//Now if I define the struct like this:
```

```
struct MyStruct
```

```
{  
    MyStruct()
```

```
{
  int i;
  for(i=0;i<5;i++)
    a[i] = NULL;
}
int a[5];
};
```

//#1 will init the struct with NULL because uses the defined struct ctor but generate almost the same bin (asm) code as case #2 and #3 (the init stack var) without the struct ctor defined!

I guess I don't understand when the compiler knows about the struct default constructor and when dose not (and expect for one defined).

Thank you
Karl M