

## Memory Leaks

by Randy Charles Morin

Victor Wagner asked me this week about retrieving memory leak information using the Win32 API functions. Definitely the question of the month and I'd like to thank Victor for the question.

You are all likely intrigued about how Bounds Checker finds memory leaks. Well, it's not really that difficult. As you will see in this article, Microsoft provided a nice set of functions for pinpointing where those leaks are occurring. A few lines of code will show you how to emulate the behavior that is provided by the hundreds of dollars per seat you are spending on Bounds Checker. My hope is that this article will convince you to save those hundreds of dollars per seat and spend them on something a little more useful.

I have used Bounds Checker on and off for many years. Generally, my interest in Bounds Checker has had little to do with removing memory leaks from programs that I've written. The most common reason I've used Bounds Checker is to show the newbie developer how useless the product really is. I start by reflecting back to my first encounter with the product. After showing my team leader that she had many memory leaks in her code, she decided that she needed a product Bounds Checker to help her create more memory conservative code. A week after the product had arrived, the team leader approached me about the thousands of memory leaks that Bounds Checker was reporting in my OWL code.

For those who had the opportunity to use OWL and Bounds Checker in combination, you likely know the problem in detail. For those who don't, I'll explain. OWL used a type of garbage collection to discard unused memory. Bounds Checker misinterpreted the OWL allocations as memory leaks. The result was that every heap allocation of an OWL object resulted in Bounds Checker reporting a unique warning. As OWL was very dependent on heap allocations, the results were quite outrageous and completely useless.

That was my first encounter with the useless product called Bounds Checker. Since then, the product has only gotten worse. In the last five years, management has told me that they wanted me to use Bounds Checker on numerous occasions. I have yet to encounter one instance where Bounds Checker has ever revealed a memory leak in my code. The reason that Bounds Checker is unable to find leaks in my code is that I've created a very small class that makes Bounds Checker redundant.

### Listing 1: FindMemoryLeaks

```
#include "crtDBG.h"

#ifdef _DEBUG
#define new new(_NORMAL_BLOCK, THIS_FILE, __LINE__)
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

class FindMemoryLeaks
{
    _CrtMemState m_checkpoint;

public:
    FindMemoryLeaks()
    {
```



cheaper. In one experience, a developer struggled for weeks with a numerous amount of tools before he finally gave into using this simple class. Five minutes later!