BoundsChecker

实验报告

课程名称：	软件质量保证与测试技术
姓　　名：	武敬信
学　　号：	202000800525
专业班级：	21软件工程1班
授课教师：	康钦马

2024 年 5 月 23 日

1 实验目的
2 实验环境
3 实验步骤
3.1 ActiveCheck 模式
3.2 FinalCheck 模式
3.3 检测 Win32 API 函数的兼容性

1 实验目的

BoundsChecker 是一个运行时错误检测工具，它主要定位程序运行时期发生的各种错误。它通过驻留在 Visual C++ 开发环境内部的自动处理调试程序来加速应用程序的开发，缩短产品发布时间。使用 BoundsChecker 对程序的运行时错误进行检测，有两种使用模式可供选择。一种模式叫做 ActiveCheck，一种模式叫做 FinalCheck。通过实验，主要掌握 ActiveCheck 模式和 FinalCheck 模式，并简单了解附带的函数兼容性检查。

2 实验环境

虚拟机：Windows XP

软件：BoundsChecker 7

CPU：AMD Ryzen 7 5800H

内存：8GB

3 实验步骤

3.1 ActiveCheck 模式

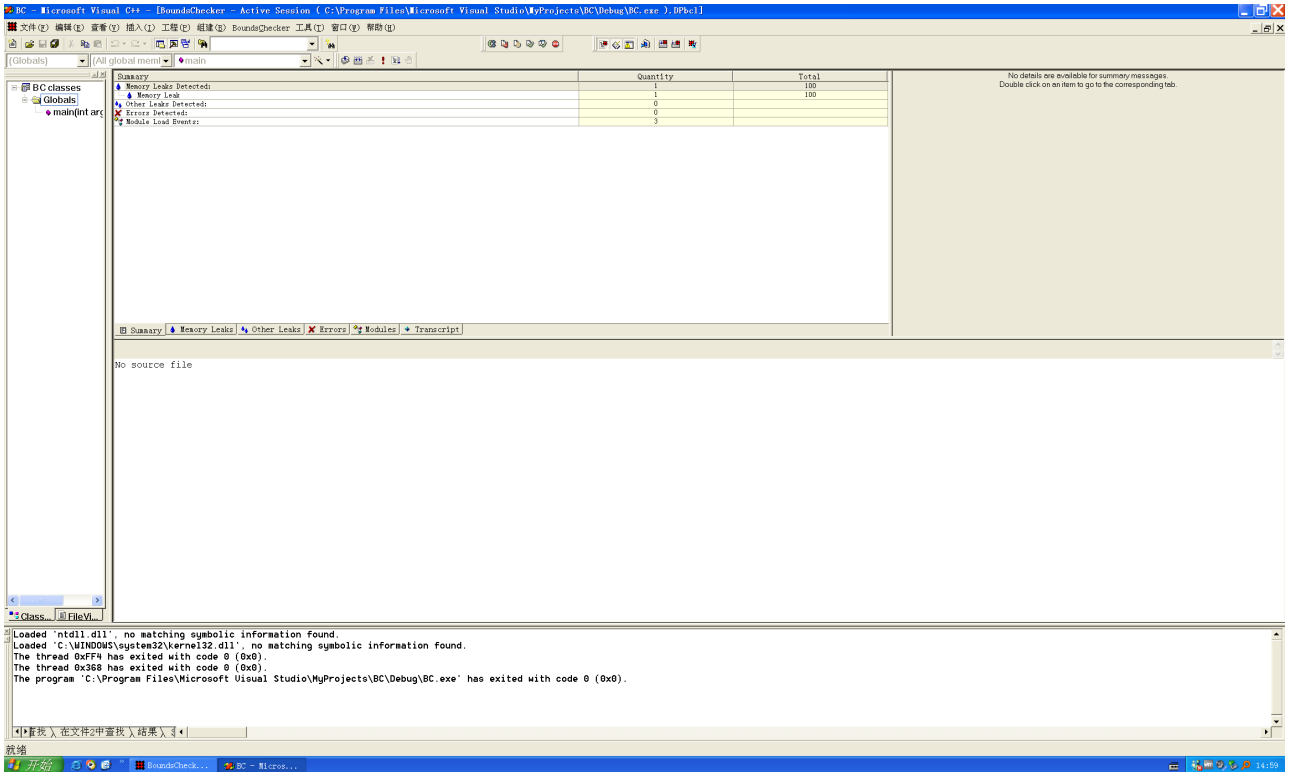
测试的代码如下：

```
#include "stdafx.h"
#include <stdlib.h>
#include <malloc.h>
#include <stdio.h>

int main(int argc, char* argv[])
{
    char* p;
    p = (char*)malloc(100);
    if (NULL == p)
        printf("Not Enough Memory!\n");
    else
        printf("Mem Allocated at: %p\n", p);

    return 0;
}
```

首先，在 VC6.0中编译这个程序。

其次，确保 VC++ 集成开发环境中 `BoundsChecker/Error Detection` 菜单项和 `BoundsChecker/Log Events` 菜单项处于被选中的状态。只有这两项被选中，BoundsChecker 才会在程序运行过程中发挥作用。

最后，在 VC++ 集成开发环境中选择 `Build/ Start Debug/Go` 菜单命令，在 Debug 状态下运行程序，ActiveCheck 也在后台开始运行了。



Bounds Checker 提供了 `Display Error And Pause` 功能，当选中的时候，如果程序执行时遇到了错误，那么会暂停执行。这一功能是否开启取决于个人喜好。

在操作全部结束后，Bounds Checker 会显示一个所发现错误的列表。

Summary	Quantity	Total
Memory Leaks Detected:	1	100
Memory Leak	1	100
Other Leaks Detected:	0	
Errors Detected:	0	
Module Load Events:	3	

在这一窗口中，会显示程序在内存、资源使用上的问题，包括种类、次数等，如果是内存泄漏，则记录损失了多少内存。

ActiveCheck 模式下，程序运行的很快，但是检测出问题的种类是有限的。

3.2 FinalCheck 模式

FinalCheck具有BoundsChecker提供的所有检错功能。利用 FinalCheck 则发现很多ActiveCheck 不能检测到的错误，包括：指针操作错误、内存操作溢出、使用未初始化的内存等等，并且，对于 ActiveCheck 能检测出的错误，FinalCheck 能够给出关于错误更详细的信息。所以，我们可以把 FinalCheck 认为是 ActiveCheck 的功能增强版。我们付出的代价是：程序的运行速度会变慢，有时甚至会变的很慢。

以下面的这个程序为例：

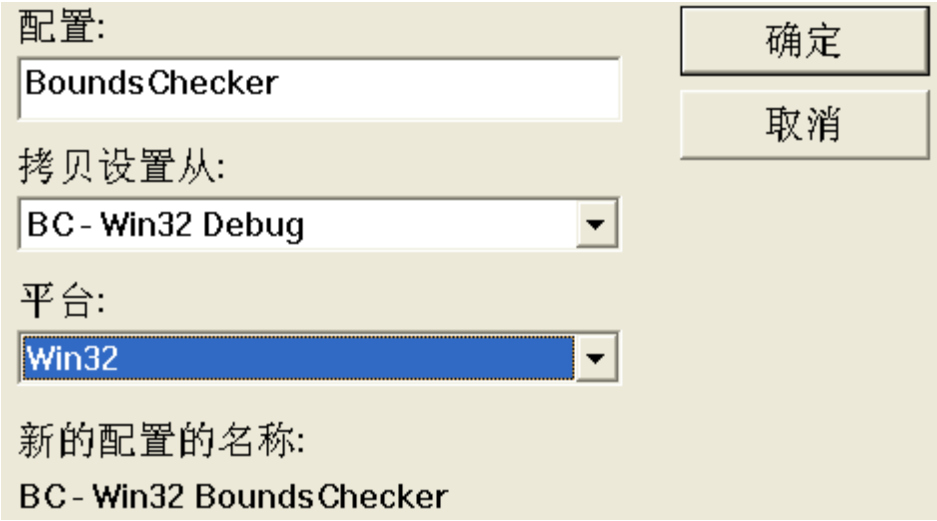
```
// BC.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"
#include <stdlib.h>
#include <malloc.h>
#include <stdio.h>

int main(int argc, char* argv[])
```

```
    int i=0;
    int a[10];
    for(i=0;i<11;++i)
        a[i]=1;

    for(i=10;i>=0;--i)
        printf("%d",a[i]);
    printf("\n");

    return 0;
}
```

我们知道，这个程序存在**数组越界**的问题，也很有可能产生内存溢出。要想在 FinalCheck 模式下测试程序，不能使用 VC++ 集成开发环境提供的编译连接器来构造程序，而必须要使用 BoundsChecker 提供的编译连接器来编译连接程序。当 BoundsChecker 的编译连接器编译连接程序时，会向程序中插装一些错误检测代码，这也就是 FinalCheck 能够比 ActiveCheck 找到更多错误的原因。

首先，在 VC++ 集成开发环境中打开你所要测试的项目。

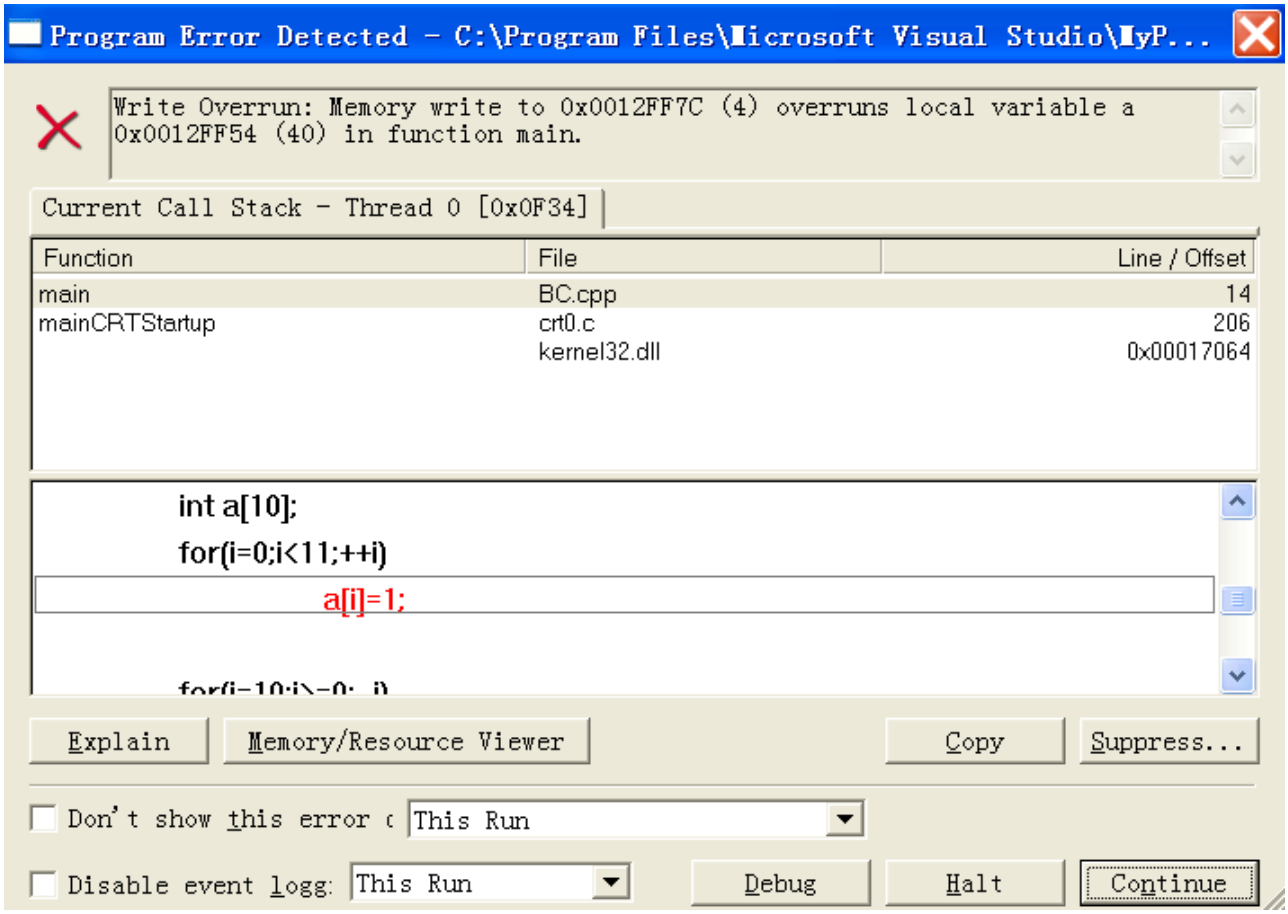
由于要使用 BoundsChecker 的编译连接器重新编译连接程序，所以我们为 BoundsChecker 独自构造一个文件夹。在VC++集成开发环境中，具体操作方法是：点击 `Build/Configurations…` 菜单命令。接着，在弹出的对话框中点击 `Add` 按钮。在 `Configuration` 编辑框中添入你为 BoundsChecker 创建的文件夹的名称，这个名称是任意的，比如我们取名为 `BoundChecker` 。



选择 `BoundsChecker/Rebuild All with BoundsChecker` 菜单命令，对程序重新进行编译连接，也就是在这时，BoundsChecker 向被测程序的代码中加入了错误检测码。编译连接完成后，BoundsChecker 会在你为 BoundsChecker 构造的文件夹中生成可执行文件。

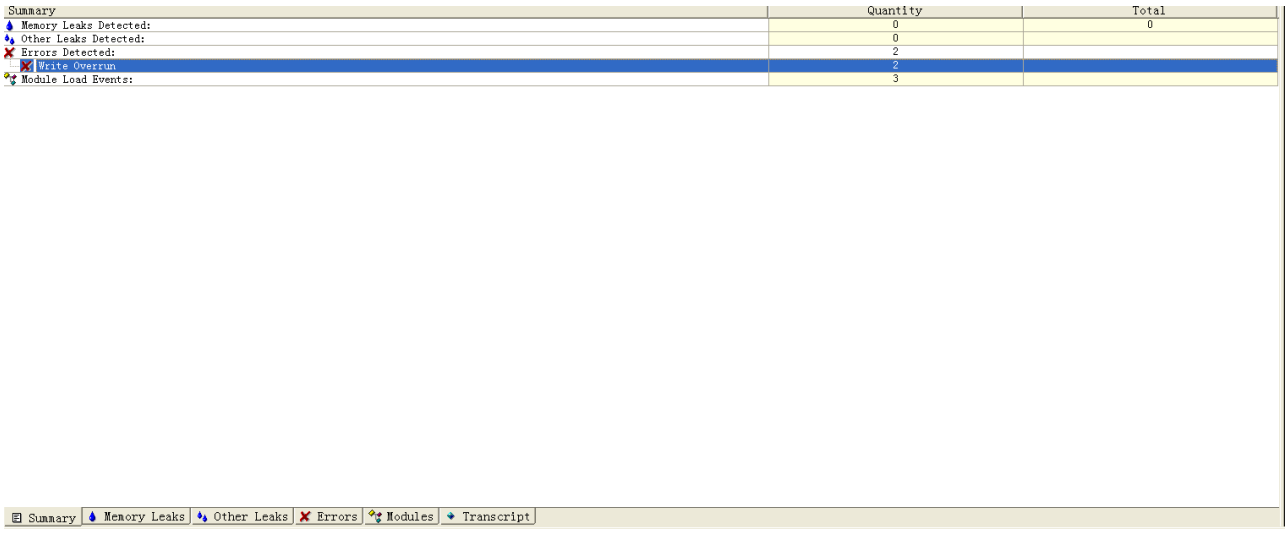
在 FinalCheck 模式下对程序进行检测的准备工作都已经做好，这时可以启动程序开始测试了，步骤与在 ActiveChecker 模式下没什么区别：确保 VC++ 集成开发环境中 `BoundsChecker/ Error Detection` 、`BoundsChecker/ Log Events` 、`BoundsChecker/ Display Error And Pause` 菜单项处于选中状态，`BoundsChecker / Setting` 中的 `Memory Tracking` 选项中的 `Enable FinalCheckt` 为选中状态。然后点击 `Build/Start Debug` 菜单，点击 `GO` 。程序开始在 Debug 状态下运行。按照你制定好的测试用例，对程序进行操作。

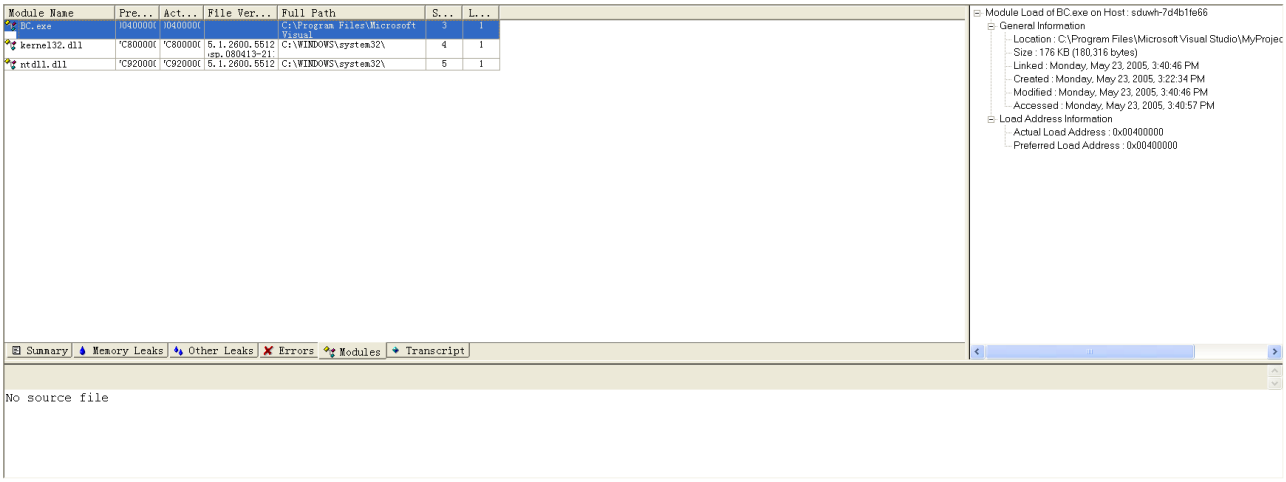
当 BoundsChecker 检测到了错误时，会弹出窗口向你汇报。



你可以当时就进行处理，也可以等到你的操作全部完成，退出程序之后再对列出的这些错误进行分析。和之前一样，这完全取决于是否选中了`BoundsChecker/Display Error and Pause`菜单项。

退出程序后，BoundsChecker 会给出错误检测结果列表。





该错误列表与 ActiveChecker 给出的错误列表的查看方法完全一样。只不过这个列表中所报告的信息会更多、更详细一些。

3.3 检测 Win32 API 函数的兼容性

BoundsChecker 还提供了一个功能——检测程序中使用的 Win32 API 函数在不同平台上的兼容性。该功能与前面提到的 ActiveChecker、FinalCheck 模式没有什么关系，它是独立的一个功能。

点击 `BoundsChecker/View/Compliance Report`，在对话框中选择程序承诺能够运行的平台，以及被要求遵从的其他标准（标准C和扩展的标准C），点击“OK”按钮，BoundChecker会给出兼容性检测报告。