

Anne Marie Heidebreicht

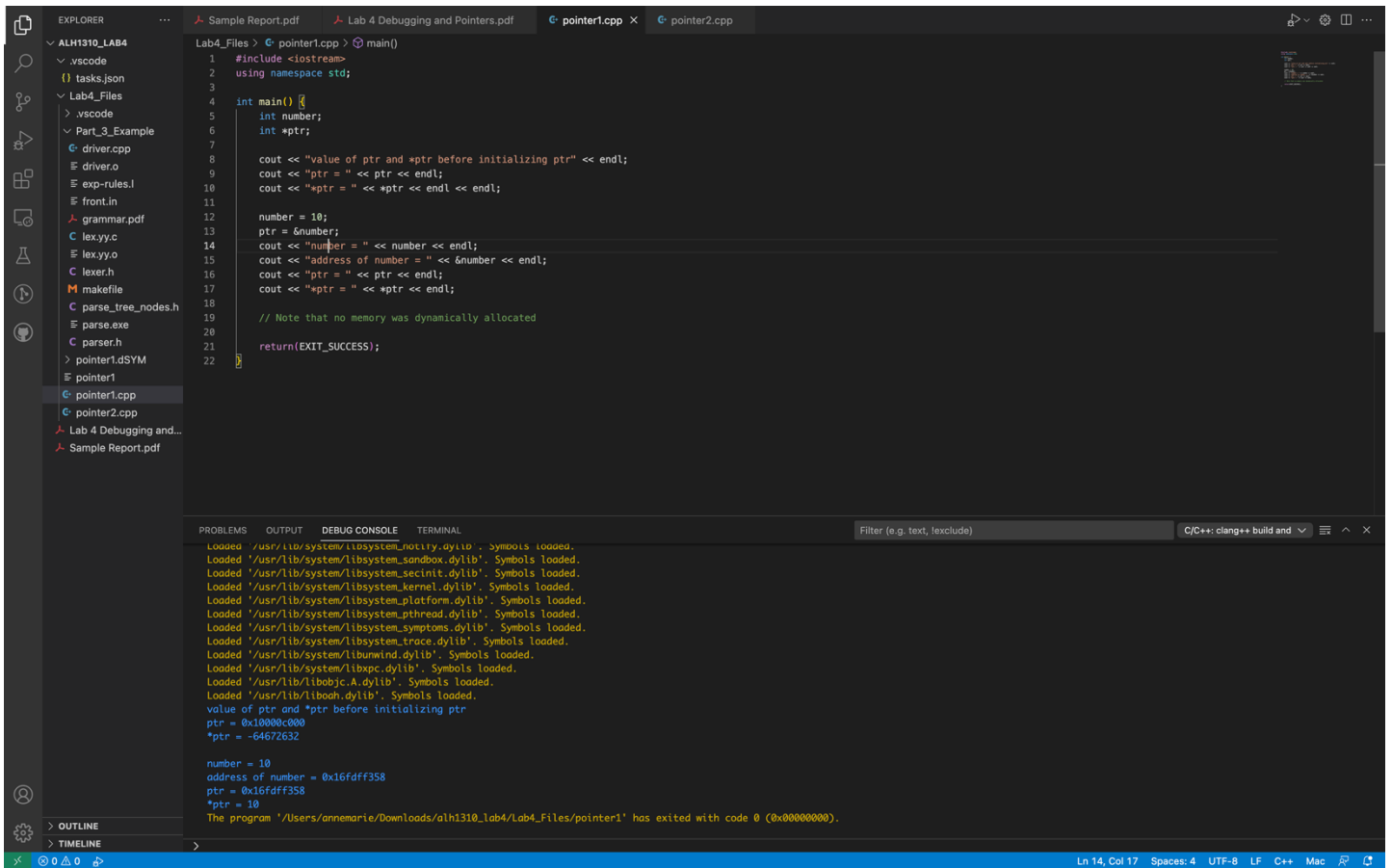
CSE 4714 Theory and Implementation of Programming Languages

April 28, 2023

Lab 4 – Debugging and Pointers

Pointer to a Stack Variable – Reference *ptr before assigning a legitimate memory address

(Resulted in printing random number for *ptr)



The screenshot shows a C++ IDE with a project named 'ALH1310_LAB4'. The Explorer panel on the left shows the file structure, including 'Lab4_Files' and 'pointer1.cpp'. The main editor displays the code for 'pointer1.cpp', which is as follows:

```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     int number;
6     int *ptr;
7
8     cout << "value of ptr and *ptr before initializing ptr" << endl;
9     cout << "ptr = " << ptr << endl;
10    cout << "*ptr = " << *ptr << endl << endl;
11
12    number = 10;
13    ptr = &number;
14    cout << "number = " << number << endl;
15    cout << "address of number = " << &number << endl;
16    cout << "ptr = " << ptr << endl;
17    cout << "*ptr = " << *ptr << endl;
18
19    // Note that no memory was dynamically allocated
20
21    return(EXIT_SUCCESS);
22 }
```

The bottom panel shows the 'TERMINAL' output, which includes the following text:

```
Loaded '/usr/lib/system/libsystem_notify.dylib'. Symbols loaded.
Loaded '/usr/lib/system/libsystem_sandbox.dylib'. Symbols loaded.
Loaded '/usr/lib/system/libsystem_secinit.dylib'. Symbols loaded.
Loaded '/usr/lib/system/libsystem_kernel.dylib'. Symbols loaded.
Loaded '/usr/lib/system/libsystem_platform.dylib'. Symbols loaded.
Loaded '/usr/lib/system/libsystem_pthread.dylib'. Symbols loaded.
Loaded '/usr/lib/system/libsystem_symptoms.dylib'. Symbols loaded.
Loaded '/usr/lib/system/libsystem_trace.dylib'. Symbols loaded.
Loaded '/usr/lib/system/libunwind.dylib'. Symbols loaded.
Loaded '/usr/lib/system/libxpc.dylib'. Symbols loaded.
Loaded '/usr/lib/libobjc.A.dylib'. Symbols loaded.
Loaded '/usr/lib/libobjc.dylib'. Symbols loaded.
value of ptr and *ptr before initializing ptr
ptr = 0x10000c000
*ptr = -64672632

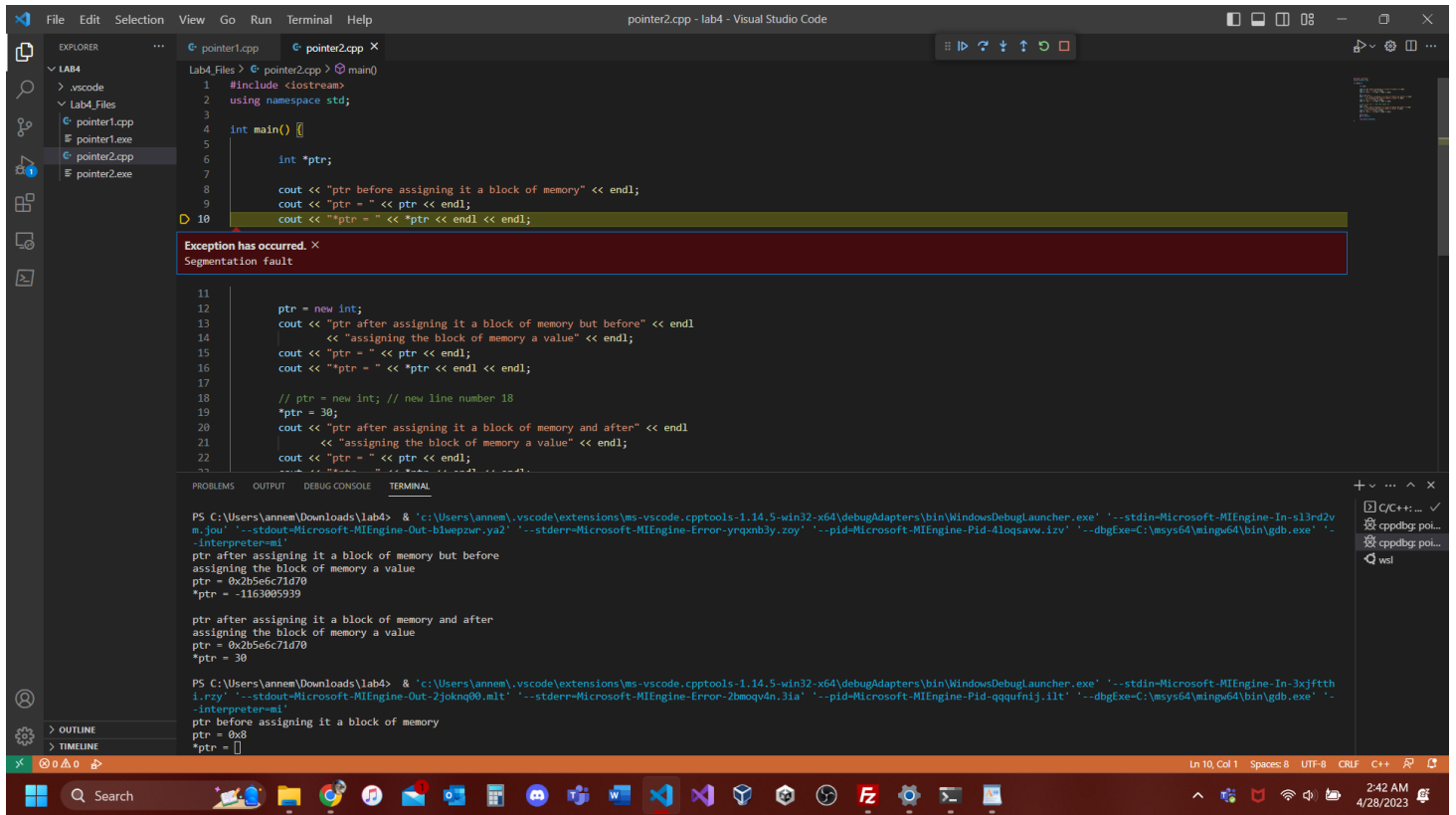
number = 10
address of number = 0x16fdff358
ptr = 0x16fdff358
*ptr = 10

The program '/Users/annemarie/Downloads/alh1310_lab4/Lab4_Files/pointer1' has exited with code 0 (0x00000000).
```

The status bar at the bottom indicates the current line and column: 'Ln 14, Col 17'. The language is set to 'C++' and the platform is 'Mac'.

Pointer to a Heap Variable – Reference *ptr before assigning a legitimate memory address

(Resulted in a segmentation fault error)



The screenshot shows the Visual Studio Code editor with a C++ file named `pointer2.cpp` open. The code is as follows:

```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5
6     int *ptr;
7
8     cout << "ptr before assigning it a block of memory" << endl;
9     cout << "ptr = " << ptr << endl;
10    cout << "ptr = " << *ptr << endl << endl;
11
12    ptr = new int;
13    cout << "ptr after assigning it a block of memory but before" << endl
14         << "assigning the block of memory a value" << endl;
15    cout << "ptr = " << ptr << endl;
16    cout << "ptr = " << *ptr << endl << endl;
17
18    // ptr = new int; // new line number 18
19    *ptr = 30;
20    cout << "ptr after assigning it a block of memory and after" << endl
21         << "assigning the block of memory a value" << endl;
22    cout << "ptr = " << ptr << endl;
23}
```

An exception has occurred, as indicated by the red bar in the editor. The message is:

```
Exception has occurred. X
Segmentation fault
```

The terminal output shows the execution of the program:

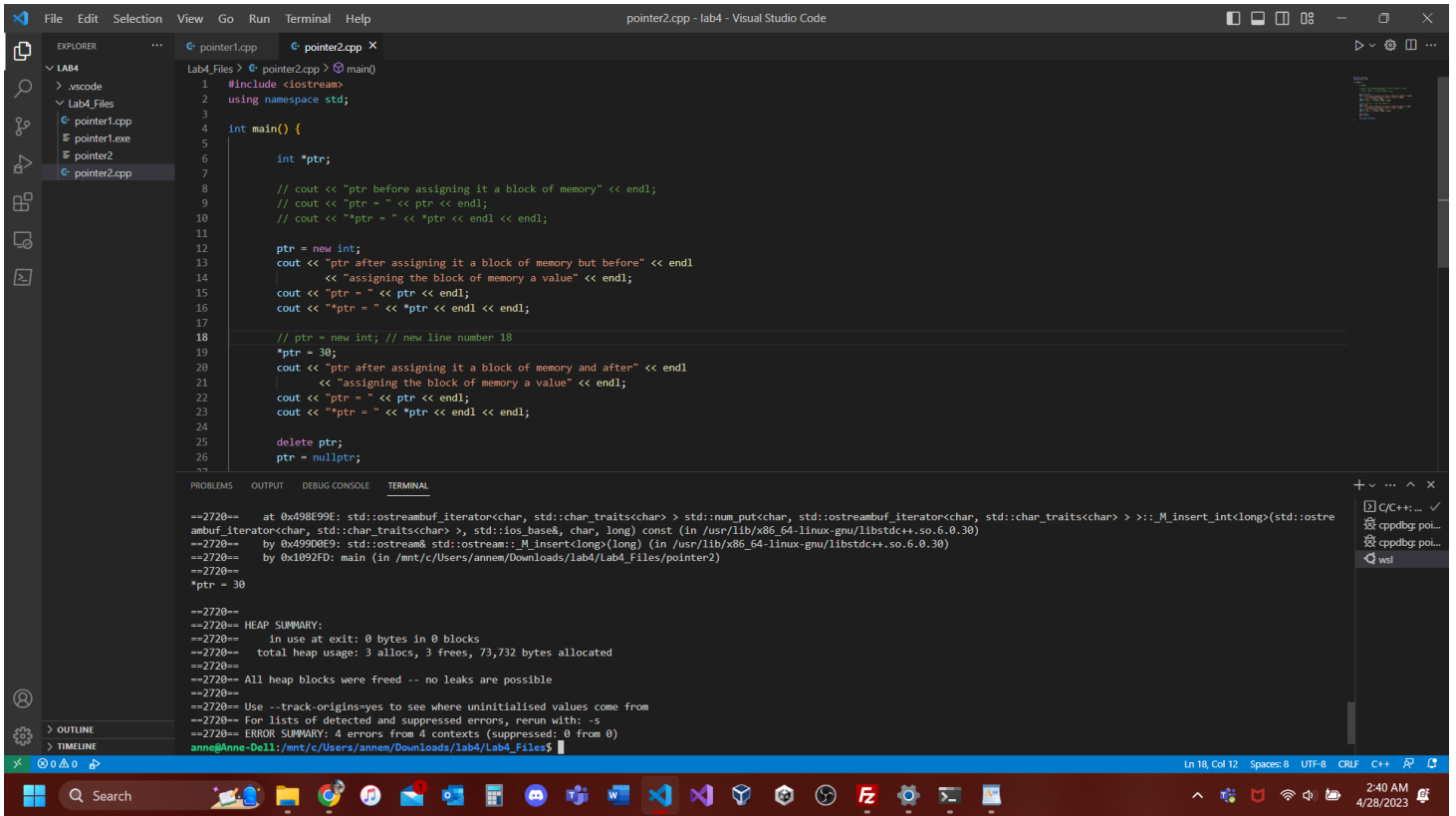
```
PS C:\Users\annem\Downloads\lab4> & 'c:\Users\annem\.vscode\extensions\ms-vscode.cpptools-1.14.5-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin-Microsoft-MIEngine-In-s13rd2v
m_jou' '--stdout-Microsoft-MIEngine-Out-blwepznr.ya2' '--stderr-Microsoft-MIEngine-Error-yrqxnby.zoy' '--pid-Microsoft-MIEngine-Pid-4loqsawv.izv' '--dbgExe=C:\msys64\mingw64\bin\gdb.exe' '-
-interpreter=mi'
ptr after assigning it a block of memory but before
assigning the block of memory a value
ptr = 0x2b5e6c71d70
*ptr = -1163005939

ptr after assigning it a block of memory and after
assigning the block of memory a value
ptr = 0x2b5e6c71d70
*ptr = 30

PS C:\Users\annem\Downloads\lab4> & 'c:\Users\annem\.vscode\extensions\ms-vscode.cpptools-1.14.5-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin-Microsoft-MIEngine-In-3xjftth
1.rzy' '--stdout-Microsoft-MIEngine-Out-2jokng00.mlt' '--stderr-Microsoft-MIEngine-Error-2bmoqv4n.3ia' '--pid-Microsoft-MIEngine-Pid-qqqufnij.ilt' '--dbgExe=C:\msys64\mingw64\bin\gdb.exe' '-
-interpreter=mi'
ptr before assigning it a block of memory
ptr = 0x8
*ptr = []
```

Pointer to a Heap Variable – No memory leak

(Resulted in 0 memory leaks found by Valgrind)



The screenshot displays the Visual Studio Code editor with a C++ file named `pointer2.cpp` open. The code defines a `main` function that demonstrates memory management using a pointer. It includes `<iostream>` and uses the `std` namespace. The `main` function starts by declaring an `int` pointer `ptr`. It then prints the state of `ptr` before and after allocating memory with `new int`. After printing the value, it deallocates the memory using `delete ptr` and sets `ptr` to `nullptr`. The output window at the bottom shows the execution results, including a Valgrind summary that confirms no memory leaks were detected.

```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5
6     int *ptr;
7
8     // cout << "ptr before assigning it a block of memory" << endl;
9     // cout << "ptr = " << ptr << endl;
10    // cout << "*ptr = " << *ptr << endl << endl;
11
12    ptr = new int;
13    cout << "ptr after assigning it a block of memory but before" << endl
14         << "assigning the block of memory a value" << endl;
15    cout << "ptr = " << ptr << endl;
16    cout << "*ptr = " << *ptr << endl << endl;
17
18    // ptr = new int; // new line number 18
19    *ptr = 30;
20    cout << "ptr after assigning it a block of memory and after" << endl
21         << "assigning the block of memory a value" << endl;
22    cout << "ptr = " << ptr << endl;
23    cout << "*ptr = " << *ptr << endl << endl;
24
25    delete ptr;
26    ptr = nullptr;
27 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
--2720-- at 0x498E99E: std::ostreambuf_iterator<char, std::char_traits<char>> > std::num_put<char, std::ostreambuf_iterator<char, std::char_traits<char>> >, ::_M_insert_int<long>>(std::ostre
--2720-- ambuf_iterator<char, std::char_traits<char>>, std::ios_base&, char, long) const (in /usr/lib/x86_64-linux-gnu/libstdc++.so.6.0.30)
--2720-- by 0x499D0E9: std::ostream& std::ostream::_M_insert<long>(long) (in /usr/lib/x86_64-linux-gnu/libstdc++.so.6.0.30)
--2720-- by 0x1092FD: main (in /mnt/c/Users/annem/Downloads/lab4/Lab4_Files/pointer2)
--2720-- *ptr = 30
--2720--
--2720--
--2720-- HEAP SUMMARY:
--2720--   in use at exit: 0 bytes in 0 blocks
--2720--   total heap usage: 3 allocs, 3 frees, 73,732 bytes allocated
--2720--
--2720-- All heap blocks were freed -- no leaks are possible
--2720--
--2720-- Use --track-origins=yes to see where uninitialised values come from
--2720-- For lists of detected and suppressed errors, rerun with: -s
--2720-- ERROR SUMMARY: 4 errors from 4 contexts (suppressed: 0 from 0)
anne@Anne-Dell: /mnt/c/Users/annem/Downloads/lab4/Lab4_Files$
```

Ln 18, Col 12 Spaces: 8 UTF-8 CRUF C++

2:40 AM 4/28/2023

Part 3 Example Code – Created a memory leak by getting rid of line 54

(Resulted in Valgrind finding a memory leak)

```
File Edit Selection View Go Run Terminal Help
driver.cpp - lab4 - Visual Studio Code

EXPLORER
  LAB4
    .vscode
    alh1310_lab4
      Part_3_Example
        pointer1.cpp
        pointer1.exe
        pointer2
        pointer2.cpp
        driver.cpp
        driver.o
        exp-rules.l
        front.in
        grammar.pdf
        lex.yy.c
        lex.yy.o
        lexer.h
        makefile
        parse
        parse_tree_nodes.h
        parser.h

driver.cpp
46
47
48
49
50 cout << endl << "**** In order traversal of parse tree ****" << endl;
51 cout << *root << endl << endl;
52
53 cout << "**** Delete the parse tree ****" << endl;
54 // delete root;
55 root = nullptr;
56
57
58 return(EXIT_SUCCESS);
59 }
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
==4703== HEAP SUMMARY:
==4703==    in use at exit: 17,362 bytes in 19 blocks
==4703==    total heap usage: 22 allocs, 3 frees, 92,114 bytes allocated
==4703==
==4703== LEAK SUMMARY:
==4703==    definitely lost: 56 bytes in 1 blocks
==4703==    indirectly lost: 376 bytes in 14 blocks
==4703==    possibly lost: 0 bytes in 0 blocks
==4703==    still reachable: 16,930 bytes in 4 blocks
==4703==    suppressed: 0 bytes in 0 blocks
==4703== Rerun with --leak-check-full to see details of leaked memory
==4703==
==4703== For lists of detected and suppressed errors, rerun with: -s
==4703== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
anne@Anne-Dell: /mnt/c/Users/annem/Downloads/lab4/alh1310_lab4$
```