Lab exercise 1

Linux environment setup, C++ review, and debugging

Due date

Jan 30, 2023

Table of content

Academic Integrity 2

Keywords 2

Introduction 2

Part 1: Set up your Linux environment 2

Part 2: Debugging a Buggy Program 3

    Debugging Methods 3

Rubric 4

Getting started 4

Some Resources 4

Academic Integrity

The following actions are strictly prohibited and violate the honor code. **The minimum penalty for plagiarism is a grade of zero and a report to the Aggie honor system office.**

- Uploading assignments to external websites or tutoring websites such as Chegg, Coursehero, Stackoverflow, Bartleby, etc.
- Copying code from external websites or tutoring websites such as Chegg, Coursehero, Stackoverflow, Bartleby, etc.
- Copying code from a classmate or unauthorized sources and submitting it as your work.
- Sharing your code with a classmate.

Keywords

Linux, C++, Debugging, Object-oriented programming.

Introduction

This exercise helps you prepare for the programming assignment by covering some introductory topics. You will set up the development environment that you will use this semester, learn how to use a debugger to debug your programs, and review the core concepts of C/C++ programming.

You are given a simple program, (buggy.cpp), that computes the area of shapes. The program has several syntaxes, compilation errors, and bugs that you should find and fix using three different debugging methods. Once you have corrected the program and committed it to the repository on the Github classroom, the autograder will grade it.

Part 1: Set up your Linux environment

This course requires a native installation of the Linux Ubuntu operating system.

Here are your options:

- Option 1 (Fastest): Create a Google cloud account using the educational credit provided here and install a VM instance of Ubuntu 20.
- Option 2: Install Virtual box on your computer and install Ubuntu on it.
- Option 3: Dual boot or native Linux environment

Important notes:

- Do not use Windows Subsystem for Linux (WSL).
- Do not use the department Linux servers.

Once your environment is set up, install relevant packages with **sudo apt install** or **sudo apt-get install**. The following packages need to be installed: g++, gcc, make, gdb, libasan5

The recommended development environment is VSCode (because it supports SSH and Git). Once installed, the recommended extensions are GitHub Classroom, C/C++ Extension Pack, Remote SSH

Part 2: Debugging a Buggy Program

Debugging Methods

There are three debugging methods in this lab.

1. **GDB.** The first method is GDB. To use GDB, you must include **-g** in your compile statement (e.g., **g++ -g -o buggy buggy.cpp**). Once compiled, you can run your program in gdb with **gdb buggy**. Once GDB is running, to run the executable, use the **run** command. If the program stops due to some error, you can run **backtrace** to see where the program stopped and the error it encountered. You can then use the **break** command to set a breakpoint before the line the error occurs and then use the **print** command to check values at the breakpoint. You can use **continue/step** to move past the break. The GDB Documentation can be found at sourceware.org.

2. **AddressSanitizer (Asan)**. (aka ASan) is a memory error detector for C/C++. To use ASan, include **-fsanitize=address,undefined** in your compile statement (e.g. **g++ -fsanitize=address,undefined -o buggy buggy.cpp**). You can then run the executable as normal, and the tool will print pertinent information about memory issues if an error occurs. The ASan tool is break-at-first-error, meaning multiple runs of AddressSanitizer may be necessary to catch all errors. Tutorials to ASan can be found here, here, and here (here is a longer video) - focus on how ASan is used.

3. **IDE Debugging.** You must install GDB, if you are using the VSCode debugger. Debugging reference for VSCode can be found here.

The buggy program starter code consists of the following:

- A struct **Point** defines a point on the coordinate plane by its x and y value.
- A class **Shape** defines a shape formed by a number of different points or vertices (created by the struct Point) on the coordinate plane.
- The function **main()** creates two shapes (a triangle and a quadrilateral) using the definitions above.

You are expected to complete the following tasks for the buggy program to be fully functional:

1. Shape's **addPoints(...)** function is expected to take in a single parameter – an unsized Point array called pts.

2. In Shape's **area()** function, the computation of variables **lhs** and **rhs** in their current state is faulty due to incorrect member access of pointers. There are two different methods to fix such member access; you are expected to use one method to fix **lhs** and the other to fix **rhs**.

3. In main(), you should **create three Point structs (corresponding to tri1, tri2, and tri3)** in **three different ways**. These points will help construct the Shape object, **tri**. You are also expected to **create four Point structs (corresponding to quad1, quad2, quad3, and quad4)** in any way you choose. These points will help construct the Shape object, **quad**. Finally, print out the area of the two shapes (tri and quad) created.

Note that your tasks may not just be limited to the ones explicitly defined above. Ensure that all dynamically allocated memory is properly cleaned up.

Rubric

Three tests check the items of the program which you have been asked to fix.

1. Compilation (34 pts)
2. Correct output (33 pts)
3. No leaking memory (33 pts)

a. Your program should compile with no errors or warnings.
b. To test your program locally on your computer, run the script **tests.sh**.

```
chmod u+x tests.sh
./tests.sh
```

c. Your program will also be manually tested and reviewed for correctness with additional test cases.

Getting started

The assignment template is hosted on GitHub classroom. Complete the following steps to get started:

1. Go to the assignment's GitHub classroom: https://classroom.github.com/a/1A23LyEH
2. Link your GitHub account to the classroom - If you cannot find your name, you accidentally linked the account to an incorrect name, or you want to change the account linked to your name, please contact a T.A.
3. Clone your assignment repository on the Linux machine
   a. Official SSH guide by GitHub
   b. This is a tutorial on how to clone, edit, and commit changes to GitHub

Some Resources

GDB tutorial: Scuffed recording on how to use GDB