Homework & Debugging Tips

Data Structures
Spring 2018

http://www.cs.rpi.edu/academics/courses/spring18/csci1200/calendar.php

Contents

Order of Consultation

Ask your question to these people in this order. Spend some time (20+ min) on each step before moving to the next one.

1. **Yourself** - The only way you'll learn to debug is by doing it yourself. Be confident!
2. **The Internet** - many people have learned Data Structures/C++ before you. (See C++ question section for some sites to visit).
3. **Your friends and classmates** - Your peers are overcoming the same hurdles as you, so they'll often have very precise advice (be careful not to share code or in-depth algorithms though – remember the Academic Integrity Policy).
4. **Mentors in office hours/lab** - If none of the above could help, the mentors have all taken Data Structures and can be a good resource. Mentors are undergraduate students.
5. **TA at office hours/lab** - TAs grade homeworks and exams. They are graduate students who might've done undergrad somewhere else, so they might not have taken Data Structures at RPI.
6. **Professor**

NOTES:

- If after steps 1 and 2 you are still stuck, you can also post on the online discussion forum.
- Steps 4-6 are a little interchangeable – obviously in office hours/lab the person who helps you is whoever gets to your name on the list and the professor's office hours are the first ones in the week.
- **This is not to say that we do not want to help you – we are here to help with what you do not understand. But these steps will help you become better at debugging your code.**

I Don't Know Where to Start

Often it's difficult to receive an assignment and know how to solve it all right away. *It's important to break up an assignment into smaller problems, solve those smaller problems,*

and then combine their answers to get the final answer (remind you of recursion?). Once you've broken up the assignment into smaller problems, solve the easiest ones first. Check your work with quick tests. Then you can work up to the harder parts.

Read the homework pdf slowly and completely, write notes about it, write down the main goal of the homework, and write down what edge cases you'll need to handle.

Discuss an overview of the assignment with your friends and classmates to clear up any ambiguities. It's okay if you don't know how to implement your solution yet, just make sure that you understand what the assignment is asking for.

Making sure that you understand the homework problem before starting can save you hours of mistakes.

It helps to think about how you would solve the problem by hand.

- Draw pictures like we do in lecture and write out the steps you need or want to take to solve the problem.
- Then translate the algorithm you used to solve it by hand into pseudocode.
- Then translate the pseudocode to C++. And, remember that simple is better than complex.

Have A C++ Question?

- Ask Google
- Check the standard library documentation on cplusplus.com
  - If you search for the function you are trying to use it will tell you:
    - Whether or not you need to have std:: in front of the function
    - What header file you need to include in the function
    - What arguments the function requires
    - If the function is C++11 specific
    - What the return is from the function
    - An example of using that function
    - The complexity of the function
- For miscellaneous questions, search Stack Overflow

But remember, always follow the Collaboration & Academic Integrity Policy that you signed. That includes listing the online resources that you consulted in the README upon submission of your homework.

Debugging On Your Own

IMPORTANT: ***ASSUME NOTHING***
Sometimes bugs in computer programs are really small mistakes:

- A '-' sign where a '+' sign should be
- An 'i' where a 'j' should be
- Compiling/submitting the wrong version of the code
- Compiling/submitting the code without saving it first

Programmers will make these mistakes. It's only human. The only thing you can do while debugging is assume nothing is right until you prove it.

That's why TA/mentors at office hours might seem like they don't trust you. They don't really trust themselves to write perfect code either. You have to test your code to prove that it works.

Rubber Duck Debugging

Ever notice that sometimes when you're explaining your code to someone else, you notice a bug that you couldn't find before? That's a common phenomenon in CS, and due to the fact that explaining it out loud forces you to mention parts that you take for granted while reading your own code. When this happens, the person you're explaining it to doesn't need to do anything!

Just the process of explaining it out loud to them was enough for you to solve your problem. Therefore, you don't really need that person to be there at all. You could explain your code to a rubber duck. Hence the name: "Rubber Duck Debugging" (https://en.wikipedia.org/wiki/Rubber_duck_debugging).

Writing It Out

Sometimes people work best when they can see the whole picture. Writing out what your code is actually doing and outputting instead of what you think/want it to do can help you figure out where the problem is. For example:

```
std::vector<int> v;
v.push_back(1);
v.push_back(2);
v.push_back(3);

for(unsigned int i = 0; i <= v.size(); i++) {
    std::cout << v[i] << std::endl;
}
```

So you might want to write out:

Vector: 1 2 3
Index: 0 1 2

For loop:
i = 0 vector size = 3
print v[0] – 1
i = 1 vector size = 3
print v[1] – 2
i = 2 vector size = 3
print v[2] – 3
i = 3 vector size = 3
print v[3] – ??

NOTE:

- When you write out your problem make sure you have found the smallest possible case to test on (otherwise you might be writing for a while)

Compiler Errors

Make sure to read the compiler message thoroughly. They can often be filled with junk and visual clutter, but usually at least point out a line in your program that they are complaining about.

After reading the error message carefully, carefully read the code around the error line. Read the code that calls the error line, and make sure that everything makes sense.

If you don't clear up your error that way, consult the internet for examples of similar errors and how other people solved them. Stack Overflow is especially good for this. If that doesn't help either, you can consult your friend or classmate, they might've seen a similar one before.

1. **Start at the first error** - find the first line that says "error" and go to the line number of that error

    a. If the compiler says:

        i. Something was unexpected or it expected something more:

            - Check the line of the error as well as the lines before the line number listed in the error
            - Check that all your curly braces ( ) match up
            - Check that all your parentheses match up
            - Check that you have semicolons in the right place

    b. If the compiler can't find something (saying it's unrecognized):

        - Is it spelled correctly? Capitalization matters. A variable called: 'count' is different from one called 'Count'.

        i. If it's part of STL:

            - Did you include the correct header file for it?
            - Did you use the proper namespacing (std:: or 'using namespace std')?
            - Did you spell it correctly?
            - If it's a function did you give it the correct type of arguments?
            - Is it specific to c++11 and are you compiling with c++11 (-std=c++11)?

        ii. If it's a class or function that you created:

            - Did you include the .h file containing the class definition?
            - Did you use include guards incorrectly? (Should be #ifndef as opposed to #ifdef, and each file should have a unique name after the directive)?
            - Do the function headers match exactly? (same return type, same parameter types, const in the exact same places)?
                – Examples of matching function headers:

```
// in the .h file:
class ClassName {
    public:
        ClassName(const std::string& s);
        void PrintStuff() const;
};

// in the .cpp file:
ClassName::ClassName(const std::string& s) {
    // constructor stuff
}

void ClassName::PrintStuff() const {
    // print
}
```

iii. If it's a class variable or function that you created:
- Is the correct .h file included?
- Is the function static?
- Did you include "ClassName::" before the name of the function when you implemented it in the .cpp file? (see example in above code snippet)

iv. If it's a local variable
- Is it in the correct scope? Is the scope of the variable too small?
  - Example:

```
for(int i = 0; i < 10; i++) {
    int count = i * 2;
}

std::cout << count << std::endl;
```

2. Once you've solve the error - save and recompile

- Sometimes the compiler will spit out more errors than what is actually there (especially if you have mismatched braces, missing semicolon, etc). So solve the first error, save, recompile, and solve the next one until there are no more compiler errors.

Linker Errors

- Look at the function it's talking about.

1. Did you implement the function (meaning have you said what the function will do)?
   a. Do the headers match exactly?
2. Is the error talking about redefining functions?
   a. Are you including a header file twice without include guards?
   b. Are you including a .cpp file instead of the .h?
   c. Have you implemented a non-member function in a header file?
      i. Did you implement a member function twice (in both the header file and the implementation file)?

- Did you use angled brackets (< >) for including the header file of your custom class instead of double quotes (" ")?

Runtime Errors

Program Aborts with an Error (like a Segmentation Fault)

- Get a stack trace (a list of the function calls that have occurred leading up to the error) – run either gdb or Dr. Memory/Valgrind/ASAN
  - Steps for running gdb:
    - gdb <executable.exe>
    - If its an assertion that's failing, type "break abort"
    - run arg1 arg2 arg3....etc.
    - – Program will halt at seg fault –
    - bt
      - short for "backtrace"
      - Will print a line with a function and line number for each function on the stack. The top one is where the error occurred, often need to go a few levels out to get out of STL functions.
      - Often the first or second line from code that you wrote will show you the issue (dereferencing null, going out of bounds).
      - If not, you can inspect the stack and select frames on the stack to look at
- Run Dr. Memory/Valgrind/ASAN – see the instructions to run on the class website under "Memory Debugging"
  - Note on running Dr. Memory with Cygwin:
    - Is there a chance you could use WSL instead? It's less trouble to run Dr. Memory on WSL (WSL also has the option of Valgrind instead of Dr. Memory).
    - In Cygwin you'll first need to compile the executable into 32-bit first (again see instructions under "Memory Debugging") using mingw
    - **IMPORTANT:** the compile line for mingw has changed depending on what version of Windows you are on.
      - If you compile with the wrong version of mingw you will see an error when you try to run Dr. Memory.
      - There are three possible compile lines for mingw:

```
i686-pc-mingw32-g++.exe -static-libgcc -static-libstdc++ -ggdb -o foo.exe
   foo_main.cpp foo_other.cpp

i686-w64-mingw32-g++.exe -static-libgcc -static-libstdc++ -ggdb -o foo.exe
   foo_main.cpp foo_other.cpp
```

```
i686-w64-mingw32-g++.exe --static -ggdb -o foo.exe foo_main.cpp foo_other.cpp
```

Infinite Loop

- Check for a single "=" where you should have a double "==" for comparison
  - The single "=" returns the value of the right hand side which is most likely definitely nonzero, and will result in the expression always being interpreted as true.
- Check for unsigned int comparisons
  - Is it possible one side of the comparison is subtracting or going below 0? If so you need to switch everything to signed integers (You can almost always safely cast vector::size's return value to an int) or try to figure out a comparison that does not depend on going below 0.
  - Are you doing math with a unsigned integer? If so the outcome of that could become an unsigned integer. For example:

```
std::vector<std::string> v;
v.push_back("Hello");
v.push_back(" ");
v.push_back("World");

int number = (5-100) / v.size();
std::cout << number << std::endl; // prints 1431655733
```

- If you eventually get an out of memory error or "stack overflow" then you could have infinite recursion
  - Check the base case of your function.
  - Use gdb:
    - Run the code in gdb, hit ctrl+C while its running.
    - bt to get a stacktrace
      - You'll probably be in some random location, but you'll at least know which loop or function you're stuck in, as well as know a trace of what parameters your recursive function might be getting.
  - Use a print statement to find out which loop is going forever
    - Run, then hit ctrl+C (or possibly ctrl+K in WSL)

    - Use file redirection when you run (using the "Misc Programming Section" of the course webpage), in order to send the output to a file if you need to see a specific print statement
  - You can use ctrl+C (or ctrl+K in WSL) to kill your program if it has an infinite loop

IMPORTANT NOTE:

- *DON'T* run the memory debugger on your infinite loop. With Dr. Memory at least if you infinite loop and use Ctrl + C to try to kill it, that will only kill Dr. Memory. Your executable will still be running – and then you have to go and find the list of running programs (using Task Manager in Windows), find your executable by name, and kill it.

It Works On Your Machine But Not Submitty

- Common issues where Submitty could differ from your machine:
  - Windows and Mac initializes variable to 0 when the code left it uninitialized.
    - Submitty runs on Linux, so uninitialized variables are, well, uninitialized. This means that when the code tries to use this variable the value of the variable is whatever value was last in that memory location.
    - Windows might not err on iterators going out of bounds of a list or vector but Submitty will

Things to Check:

- Have you fixed ***ALL*** warnings (except maybe those in the provided code)? Compile with -Wall
- Have you changed the provided code at all?
- If it's a test case where you are shown the command line arguments used – what does the output look like on your computer using the exact same command line arguments?
  - Make sure you have not changed the provided input file
- Do you have strange characters in your code?
  - This one can be hard to catch. It can happen sometimes if you have different keyboards on your computer and you accidentally switch between them
- Have you run Dr. Memory/Valgrind/ASAN on your machine?
- Is there a possible corner case your code might not cover?

The Output Is Wong

- Things to check:
  - Have you fixed all warnings?
  - Have you changed the provided code?
  - Have you changed the provided input file(s)?
- General guideline to figuring it out (after asking the above questions):
  - What is the smallest case where your program will produce errors in the output?
    - Are there multiple small cases where it doesn't work – is there anything common among those cases?
    - If the problem is on a larger case – is there a way to make the larger case smaller?
  - What is your code actually doing vs. what you think/want it to do? (a good idea is to write out what is happening line by line, if the case is small enough or if you can focus on a specific section of the code)
  - Use gdb to observe variables
  - Use print statements to see indexes, the size of your vectors/lists/etc., the value of variables, etc.
    - Notes:
      - Say what it is you are outputting so that you know what is what if you have a lot of print out statements
      - Add std::endl to the end of your debug print statements so that the output doesn't mash together

```
std::cout << "i: " << i << " vector size: " << v.size() << std::endl;
std::cout << "vector element at i: " << v[i] << std::endl;
```

      - Remember to delete your debugging statements when you submit your homework to Submitty

Memory Leaks

- Make sure to compile with -g so that the memory debugger will show line numbers
- The memory debugger will tell you which memory isn't being deleted - it will give you the line number where it was created.
  - This doesn't mean that the exact problem is on that line – you have to find where the delete is supposed to be placed
- Is the destructor cleaning up memory?

- Is your assignment operator cleaning up the memory that was in the object being assigned to before copying in the new values?
- Is there any other place where a pointer is being set to a newly allocated array but the memory the pointer was previously pointing to wasn't deleted?
  - Note:
    * It's always a good idea to set a pointer to null after deleting it. For one thing, deleting null is defined as a no-op (aka it effectively does nothing, but it's not an error or segfault), while deleting memory that was already deleted is an error.
      · Also, this lets you debug memory leaks because it allows you to check whether a pointer is null before you point it to new memory. If it isn't, you'll be leaking that memory because if that memory were deleted you would no longer have a pointer to it.

Asking a TA or Mentor

About an Error or Problem in the Code

- Ask your question or state your problem right away.
  - Often students start with "Here's what I've done..." and try to explain their entire program before asking a question; just ask! This will speed up the process, and allow you and others to get more questions answered
- Don't just shove your code at the TA/Mentor – explain what you have done and why after explaining what your problem is, also explain what you have tried to do to debug it
- If you have run gdb/Dr. Memory/Valgrind/ASAN – show the TA/Mentor what the output is
- If the output is different from Submitty, show the TA/Mentor what the difference is

About How to Start

- Try not to just ask "Where do I start?" and leave it at that. It's a valid question, but it can also be a large question. Try to give us a little more detail so we can figure out exactly what you need to get started.

  - What part of the homework do/don't you understand?
  - What do you know you need to do?
  - Is the problem that you know what you need/want to do but don't know exactly how to write the specific C++ code?
  - Is the problem that you don't know how to break up the homework into smaller chunks?
  - Are you having trouble figuring out an algorithm for a certain part of the homework?

- If a TA/Mentor is explaining a rough algorithm to you: **don't try to convert every sentence said into working code right then and there while the TA/Mentor is still explaining.** You'll likely miss part of the explanation as you try to multitask figuring out syntax, typing, and listening.
  - Understand why you might want to do something a certain way - ask why if you don't
  - If you are worried about forgetting what the TA/Mentor is saying ask them to repeat it while you type out an outline in plain English for yourself to follow.