A Structured Programming Alternative

Eli Richmond

What this document is and isn't

This document is to explain the basic concepts of this idea. So when I present this - and the things not addressed here - in person, each of you will have at least a shallow understanding of these concepts.

This document does NOT address why this is important. This document does NOT address why I think our old system could be better. This document does NOT address potential issues with these ideas. This document is NOT meant to be persuasive.

I deeply appreciate each of your criticisms. I am completely open to any ideas that could make this better. Thank you.

What is this class for?

- ➔ The Intro to computer science class is for explaining why and how programming is useful and beautiful.
- ➔ It is for giving the most accurate experience possible of programming in the real world.
- ➔ It is for getting a small taste of everything the students will continue to expand on in later computer science classes.
- ➔ And finally, it is for understanding how the fundamental concepts of programming work NOT memorizing how they work.

But so what, what actual realistic changes can we make to account for these? First let's clarify who actually needs to take this and the basics of how this class would work.

Where does it fit in the degree plan?

This class would count as a credit in place of Structured. However, Structured needs to stick around because it does a have a place. For example, electrical engineering majors would probably only take Structured because the time commitment is lower. Also, they don't need an accurate experience of what programming is like in the real world. And they don't need to get a taste of what a computer science degree is all about. Ideally computer science majors would take this instead of Structured, but of course that is up to them. It is optional. So again, what real things can we change?

Everything the professor isn't involved in - The 25%

I would say that 75% of what makes a class absolutely fantastic is how much care, effort, and enthusiasm the professor brings. Even if a class has perfect homework assignments, test format, and covers the right material, it hardly matters if the professor doesn't care about the class. The course will be ineffective and uninspiring. I really wanted to address the solutions to this 75% first, but to understand those solutions better we need to understand how we can fix the first 25%. I.e. course material, homework, tests, labs, and grading.

(Each suggested fix is ranked by its importance. Since we are starting with the less important stuff, the rankings reflect that.)

#5 This class will cover almost every topic that the student will take in a course later.

An Intro to Computer Science class is all about helping the student determine whether or not they want to continue in this major. We should expand our course to cover bits of everything that they will continue to learn throughout their degree. In Harvard's CS50x class, they are covering everything we learned in Structured in four weeks. Let me say that again, they are covering in 4 weeks what takes us 15 weeks. Let me give you an idea of what I mean by the topics that they will take later. Here I'm reading the topics directly from the 7 extra weeks that are given in the 11 week CS50x class.

- Memory
- Data Structures
- Python
- SQL
- HTML, CSS, Javascript
- Flask
- Final Project

#6 The homework will be harder and take longer

First of all let's keep in mind that Structured doesn't have any homework assignments which is just ridiculous. With our current degree plan the entire first year for a computer science student is completely wasted because they still have no idea of what real programming is like. And they also don't even know what the future of their degree will be about (Referring to #5). So for now let's assume the homework I'm talking about is on par with the homework we do in Object Oriented Programming. The homework I've encountered so far has been okay, but far from great. It is very hard to write good homework problems, so I am not too upset that these aren't optimal. A lot of our homework lacks a hard problem that underlies the actual computer science stuff. Half of programming is thinking about hard problems. The best homework assignments are just the right amount of challenging, interesting, and relatable. But again, the main problem for Structured Programming's homework is that it doesn't exist.

#7 Open book tests & no more labs

Closed book tests clearly contradict what Intro to Computer Science is for. They focus on memorizing programming concepts and studying to memorize in order to pass those tests. That takes time away from actually solving computer science problems which is far more important. But let me quickly clarify what I mean. By open book I mean the syntax for C++ should be available during the tests. We should be graded on our ability to use a for loop in the context of a problem, not be graded on whether or not we memorized the exact syntax. Anything worth memorizing is worth looking up. Labs should go too. They aren't for thinking about problems or making you a better programmer, so why are we doing them? Are they to help teach programming? If there is something the student needs to learn, it should be demonstrated on a live example in the lecture. Labs aren't even effective for learning. As a student, I just want to get them over with because they are so boring and uninteresting. It is more about optimizing our time, and again, that time is better spent pursuing real programming problems which aren't boring or uninteresting.

#8 Grading should be pass / fail

The students taking this class should not have their GPA punished for exploring whether or not computer science is for them. Structured is actually the worst of both worlds because their GPA can be messed up, and they don't actually learn what programming is like. David Malan did a research paper that showed more students gave computer science a chance when they switched to pass / fail grading. The paper also shows that the pass / fail students spent more time on the course than their letter-graded classmates. I also can enjoy, appreciate, and evaluate a subject better when the pressure of my GPA isn't there. There is nothing like the blinding threat to a student's GPA to suck all of the awe and wonder out of a subject.

#9 Automated code grading

Come on this is an absolute no brainer. We are the computer science department and we still don't have homework grading nearly 100% automated where appropriate. We are wasting more resources than we should due to this inefficiency. Third party software isn't a good solution either. Look what happened with Mimir.

Everything the professor is involved in - The 75% (mainly lecture)

As I mentioned earlier, this is the most important idea in this proposal. The enthusiasm and fire a professor brings to the classroom accounts for 75% of what makes a course really fantastic. But really great professors are rare. Statistically most professors are

average. That is normal, but our current system doesn't take advantage of our great professors enough. This is a really hard problem.

To help explain my ideas let's assume a mock scenario. We have three Intro to CS classes this semester each with 20 students enrolled. The schedule for this class is 2 hours and 50 minutes of lecture on Monday; although, most lectures will only take roughly 2 hours. Then 50 minutes of in class time on Wednesday and Friday. Also, attendance is optional. This might seem a bit confusing, but as we go through these next four points it should make sense.

#1 Let our best lecturers lecture to everyone

In our hypothetical Intro to CS, the best lecturer out of the three professors will give the weekly lecture to all 60 students on Monday. This gives every student the best lecture we can reasonably give them. Also, this stops the duplication of effort by consolidating three separate lectures. Now to be clear this isn't your typical off the cuff lecture. This is intense, efficient, exciting, and carefully thought out. This must be our best effort. Believe it or not, enthralling lectures are possible.

#2 Record the lecture

I strongly believe every course at Astate should be recorded - even if it isn't a high quality video. There are so many benefits. I.e. referencing it later, watching it at 1.25x speed, rewinding if you didn't understand something, and actually listening and thinking vs taking notes. Also, we can't improve on something we can't remember. Next semester's lecture can't be better if we can't reference and critique the last one.

#3 In class time is used for personalized help

In class time should be used to get elaboration on specific topics, ask questions, work on the homework, and just connect. Thomas, a TA in my first year experience class, told us that by the time he got his bachelors his entire class didn't know each other's names. This shows the crazy lack of connection with our current system. Dedicated time like this will also take some strain off of our professor's email box.

#4 The non-lecturing professors should improve the course & make "shorts"

Since two out of our three professors aren't giving lectures, they have an extra 2.5 hours (plus lecture prep time) a week. This time should be used to make the course better. Try to find something that sucks. Fix it. This might be writing a better homework problem or

thinking hard about critiquing the syllabus. Also, it might be helpful to go ahead and make "shorts". These are bite sized clips that address specific programming topics. Everything should be taught in the lecture, but sometimes it helps to hear it in another way.

It is about more than the actual changes

This experiment is so important. Trying new and risky things is the only way to improve anything. Getting the greenlight to run this test means so much more than the test itself. It nudges our culture in the right direction. It lets students and faculty know they can and should make things better. I could be completely wrong, or perhaps just too critical. Maybe we are already doing the best we can do. But every classmate I talk to feels their CS education isn't getting the care and effort it deserves.

Thank you.