My Programming Journey

Zero Contradictions

October 18, 2023

Contents

1 Early High School

Before I attended high school, I was always scared of using computers. They seemed too complicated for me to figure out how to use, they seemed to crash often, and my only experience with them was limited to computer games that I had played and when I learned how to use Microsoft Office in middle school. In middle school, I had no idea what career I wanted to do when I grew up, although I had a sense that I wanted to do get a white-collar job related to technology. I was clearly smarter than nearly all other kids my age, and when computer science was explained to me as "problem-solving", I had suspicions that it may be a good career after all.

I didn't overcome my fear of computers until I went to a charter high school focused on technology in ninth grade. I hated my first month there because I was falling behind in the computer courses, my school-assigned laptop had wifi problems, and sometimes I couldn't figure out how to use my laptop to complete my school assignments. I wished that my father could've homeschooled me or sent me to a different high school. I persevered through it anyway, and I got caught up in my assignments once my laptop got fixed and I took extra time to complete the assignments.

For my technology class, my first assignments were to write basic HTML webpages. As the school year passed, we gradually moved onto learning about the history of the Internet, how to use Adobe Photoshop, electrical circuits, writing research papers about Mars, etc. Eventually, my peers and I were taught really basic programming concepts with Alice (a virtual programming language) in ninth grade. I thought it was really fun, but I knew that it was hardly close to "real" programming at all. Unfortunately, the school had a policy that students could not be taught how to do "real" programming until tenth grade. I didn't like this, and I didn't want to wait a year to start learning how to use a real programming language like C or Java.

Later that year, my parents bought me a book called *Learn C++ in 24 Hours*. It is frankly a horrible book for learning how to program compared to all the other books out there, but nobody in my family knew any better. I suppose it was better than nothing, but there are several problems with the book:

- It is misleading to make people believe that you they learn an entire programming language in just a single day. But since I was only 14 years old at the time, I didn't know any better.
- A majority of the supposed benefits that it represents about C++ are vague, and are applicable to all the other programming languages in the C language family.
- It didn't teach how to access library functions, which is the bulk of the actual language. No introductory teaching of a programming language can be considered complete if it doesn't even teach how to access library functions.
- It wastes time talking about things that don't really improve the quality of programs, like OOP, operator overloading, etc.
- The *C Programming Language* by K&R is a better book for introducing programming. It explains many concepts better than *Learn C++ in 24 Hours*, including many topics that it doesn't cover at all, like pointer arithmetic, library functions, structures, the UNIX system APIs, etc.
- C++ isn't a very good language for teaching programming in general. It's better to learn C before learning C++.

Although my high school provided me with a school-assigned laptop, students were never allowed to take these laptops home because they were school property. Since I was reading *Learn C++ in 24 Hours* at home, I wrote my first computer programs on an iPad because that was the only personal computer that I had at my house. I didn't even have a physical keyboard to type on since my parents couldn't afford to buy me a real computer and we lived paycheck to paycheck. Due to the iPad's restrictions, there was no way to download a C++ compiler onto it. The only work-around that I could find was to use an online C++ compiler that could run in the iPad's Safari web browser. Sometimes the work that I typed into the online compiler didn't get saved, but sometimes it did save the code as web browser cookies. I only typed short programs due to the limitations I had. Eventually, I finished reading the entire book and the school year ended. Much to my surprise, I wasn't afraid to use computers anymore at all, and I was enthusiastic about being a computer scientist.

2 Late High School

In tenth grade, I transferred to a better high school. Unfortunately, the new high school didn't offer any programming classes that I could take during tenth grade. Around this time, my grandmother offered my parents some money to buy me a computer. I was gleeful to finally have my own personal computer, but it still wasn't the one that I wanted. My father was/is an Apple fanatic, so he insisted on buying a 15-inch MacBook Pro, even though I wanted a computer that could run Linux instead. The Macbook Pro's environment wasn't very easier to tinker around with, and I was also frustrated because I couldn't find any IDEs or text editors for learning how to program that felt comfortable for beginners (myself in particular). Many of the online Linux tutorials couldn't be followed on MacOS. In hindsight, I was right that this held me back.

When I got my first Macbook Pro, I thought I could master all the programming languages out there by downloading syntax reference sheets (since I had the very mistaken belief that knowing programming language *syntax* was equivalent to knowing the entire programming language). Although I always told myself that I would learn programming and how to become an effective computer scientist on my weekends, I never really did. I had untreated ADHD and I didn't know where to start. Instead I wasted all my time reading stupid polls about the best operating systems and the best programming languages. I really wasn't learning anything.

Before my junior and senior years, I tried to get into electives that would teach programming, but my high school only offered one class taught any programming. Unfortunately, I couldn't take the prerequisite courses for that course, since I spent ninth grade in a different school and had to catch up on the new school's graduation requirements. In general, I'm sad that I wasn't able to get any of the electives that I really wanted during high school. The electives that I did take during high school were mostly a waste of time.

In my junior and senior years as I became more interested in linguistics and politics, instead of my deluded pseudo-progress on learning actual computer science. As I became interested in Esperanto, the Indo-European language family, and linguistics in general, I re-considered that I may want to do linguistics as my life's career instead of being a computer scientist. I could've saved myself so much time if I had been introduced to philosophy at an earlier age, and pondered its implications for how I should live my life and organize my theory of value. I didn't know how to learn fast and effectively.

3 College

College was a huge game-changer for me. New concepts and new assignments were being thrown at me so fast, and I was learning more efficiently than ever before. The speed at which I was being taught and forced to learn new things (that had been inaccessible and unknown to me in prior years) made me realize just how ineffectively I had been using my time during my teen years. It all opened my eyes to realizing just how much knowledge there is out and what kinds of things there are to learn.

In my freshman year, I started out double-majoring in computer science and linguistics. I was told from the beginning that if I could pass all the courses required for the CS minor, then I would have enough qualifications to get an internship at one of the city's local tech companies as a software developer. This made me want to complete all the required CS courses as quickly as possible. Unfortunately, everything didn't turn out as I had planned.

Throughout my first CS course, I was taught a combination of boring stuff that I was already familiar with (e.g. variables, if statements, while/for loops, etc) and stuff that I had never dealt with before (e.g. Eclipse IDE, Java syntax, library functions, etc). Even though I knew more about computer science and programming than most of the other students when I started out, all the other students were more eager to spend their free time playing with Java, were better able to keep up with the class-assigned textbook, able to complete the programming projects faster than I could. While I thought the new content that was being taught was fun, I prioritized the class last since I was prioritizing my other classes (Calculus II, Phonetic & Phonology, Technical Communication, etc) first. Although I managed to complete all the assignments and learn everything that all the other students learned in the first CS course, I still only got a C+ because half of them were turned in late.

I spent the summer after my freshman year getting my driver's license, getting my first job, and reading/thinking about philosophy for my first time. Ideally, that stuff should've been done while I

was in high school, but my mother couldn't supervise me when I had my learner's permit because she had/has terrible anxiety issues, and it was way too stressful to drive with her constantly screaming at me about the most unreasonable things regarding my driving. My father couldn't teach me how to drive either because he was busy working full-time until his work schedule had changed earlier that year. Since I didn't have a driver's license and I had untreated ADHD, there were significant barriers for me to get a job during high school. As a consequence, I didn't have any time to improve my programming skills before the next semester because I was busy completing these objectives instead.

I took my second CS course and discrete mathematics during my sophomore year. At this point, I had finish two calculus courses, and all my peers knew as much computer science as I did, except that I was more familiar with Unix and Vim due to prior experience. I once again found it difficult to keep up with the programming projects as the semester progressed. A lot of this was because I didn't like how the courses were being taught, and I thought it would be easier if I tried to learn computer science by myself since I now had a better idea of what I needed to learn. Halfway through the semester, I changed my major to mathematics since I was struggling to keep up the programming project assignments. I found the mathematics courses to be more enjoyable, and I noticed that although I was consistently getting better grades in my math classes compared to my CS courses, most of the other CS majors got better grades in their CS courses than their math courses. I couldn't figure out why there was such a discrepancy, but at the time, I figured that I probably could've been just as good at programming as they were if only I had more time to study and practice these concepts at my own pace. I was still planning to finish all the CS courses required for the CS minor because I wanted to get an internship doing software development.

Unfortunately, I didn't do very well in my third CS course either (systems programming and the C programming language). I enjoyed it the most out of all the CS courses that I had taken so far, but I still struggled to turn the programming projects in on time because I focused most of time on completing my mathematics courses, which I thought were more interesting. For the CS minor, I was also required to take five one-credit "hatchery" courses for learning CS skills that are important in the software industry (social justice, Unix/Linux, version control and Git, agile development, and databases and SQL). The social justice course was a waste of time, but I did pretty well in all of those courses, and I once again thought that I would've learned all this stuff if I had learned it all at my own pace.

At the end of my sophomore year, I had a really bad feud between me and my parents. I couldn't stay at their house, so I signed up for summer classes so that I could qualify to stay at on-campus housing at my university over the summer. During this summer, I finished the last two one-credit hatchery courses, and I retook the system programming course to pass it with an A. The last course that I had to finish for adding the CS minor to my college transcript was data structures. While I really didn't feel ready for it, I took it anyway because I had to be taking at least 6 credits that summer in order to qualify for staying in the university's on-campus housing, and there were no other courses that I could take that summer that would count towards my degree plan since I had already finished all the lower division courses. Thus, the alternative to not taking the data structures course was that I would become homeless until the feud between my parents and I cleared up.

As expected, I wasn't ready for the data structures course at all. I bombed it, and while I understood pretty much all the data structures that were taught and was doing decently well on the pencil and paper exams, my programming skills were still terrible. If the feud between my parents and I had never happened, I could've stayed at their house for the summer while honing my programming skills to finish passing all the courses for the CS minor, perhaps even to find out that I probably don't want to program for a living after all. That would've been a much more productive summer for me.

I continued to pass my mathematics and linguistics courses during my junior year without any problems at all. At this point, my plan was to just finish college with my two bachelor's degrees and CS minor, and then spend some free time after graduation to gain the skills and knowledge for becoming a software engineer. I tried to retake the data structures course again because I had to fit it into my degree plan's schedule *before* I graduated, but I failed it a second time, which was even more disappointing because I would've passed the course if only my grade were 5 points higher (i.e. if I had turned in a programming project). Once I had failed the course a second time, I gave up on trying to finish the CS minor altogether. With everything being considered, it feels like that I learned how to program the worst way possible.

I started reading about neurodivergence at the end of my junior year. I knew that I was first

diagnosed with ADHD when I was 8 years old, and I wondered if this could be a factor in why I had trouble turning assignments in on time for my college courses, so I consulted my doctor. Contrary to what my parents had told me, my ADHD was not "cured" and my condition would never go away since much of it is rooted in how my brain has a different structure compared to most neurotypical humans. After talking with my doctor, I was prescribed me a medication to help me finish college.

Unfortunately, I wasn't able to use the summer after my junior year for any personal development since I had more familial issues to deal with. My parents had never cleaned their house within over a decade since they're lazy and incompetent (by my standards anyway). Almost every room had documents, junk mail, and papers that had accumulated over the years, and all of it had to be sorted to figure out what to keep or throw away and there was lots of junk and debris around the house in general. Every day I took the Vyvanse pill, my mind automatically felt compelled to clean and organize the house as much as possible. I had never experienced any similar qualia in my life before for such sustained lengths of time, but it felt amazing. Everybody was surprised that I was able to clean so much single-handedly by myself. By the end of the summer, the house was as tidy and organized as it had ever been since my family had ever moved into it. Coincidentally, my maternal grandfather had to move into our house that summer since he was terminally ill. Although no one was planning on this, it's a good thing that I cleaned the house (with perfect timing), otherwise there wouldn't've been enough room to have him move in with us during his last days.

3.1 Miscellaneous Things

When I was 15 years old, I willingly switched my keyboard layout from Qwerty to Dvorak, then to Colemak a few months later, so I learned how to type on a keyboard three times over. When I first heard of Emacs, I knew that it had a *really* high learning curve, but I also knew that that was exactly what I wanted to use as my text editor, based on the online testimonies of people who use it. Unfortunately, I was mislead into believing that I couldn't use Emacs on the keyboard that I had due to a phenomenon known as the Emacs Pinky. While this is true to a great extent for vanilla Emacs, this isn't true for Spacemacs, which first came out in 2015. It tends to use the spacebar instead of the Ctrl keys as much as possible.

I was also mislead into believing that I couldn't use Vim because most of the Vim shortcuts are positioned for a Qwerty keyboard, and I couldn't figure out a way to map the escape key (<Esc>) closer to my fingers. In actuality, neither of those actually doesn't matter, and it's possible to use Vim regardless of what keyboard you're using, but I didn't know this. I started using Vim when I was almost 19 years old, and I switched to Spacemacs a year later when I was 20. I never got proficient in Emacs until late 2022 when I switched from Spacemacs to Doom Emacs. Doom Emacs is even easier to use and customize, and it's also faster.

In college, the courses taught Java and Eclipse. Maybe it's just me, but Eclipse never felt "right" to me, and that's one of the reasons why I was slow at completing the assignments for the CS classes. It's probably related to my Autism and OCD, but Emacs is the only computer interface that feels "right" to me. I know that Richard Stallman (the founder of the GNU Emacs project) definitely has autism as well.

In any case, I probably would've decided that I didn't want to program for a living if I had managed to become proficient at Emacs at an earlier age. Emacs is most commonly used for writing and programming, but it can do almost anything. I personally use it to journal, manage my todo list, write my webpages, play music, and use my command shell, among other things.

4 After Graduating College

A year later, I graduated from college, and I now had a lot of free time. I was determined to do what it was necessary for me become a software engineer, and I thought that I was going to finally have great success this time since I could finally learn at my own pace, I had treatment for my ADHD, and I could finally get the perfect computing environment that I had been longing for. My previous efforts in programming were hampered to a great extent by not having a good environment for doing programming that was comfortable enough for my autistic needs, so I bought a great ergonomic keyboard, a decent computer that could run Linux, and I finally became proficient in Emacs. I also read some books and videos about programming and computer science (e.g. *Out of the Tar Pit*, *Computer Organization and Design* by Patterson and Hennessy, *Theory of Computation* by Sipser), and I learned that contrary to what had been proselytized to me all these years, Object-Oriented Programming actually has a lot of disadvantages. At this point, I thought I had no excuses for not becoming a software engineer and acquiring the same amount of knowledge that someone with a bachelor's in computer science has.

Although I had been trying for months, I just didn't find programming to be as interesting as I thought it would be. I thought about trying different programming languages, different specializations, but I couldn't make myself enjoy it, especially compared to how much I enjoyed other subjects. Ever since I was 14 years old, being a programmer or software engineer were the main careers that I had in mind for my future, although I had also considered becoming a linguist, a data scientist, or a mathematics professor who does research in academia as well. I became black-pilled on doing any research in Academia once I realized that most academic research is fake, however I still believe that I would've had a better shot at becoming an academic researcher if I was given better guidance and more advanced mathematics education earlier on). For the longest time, I also believe that the best path towards having a high income and good job security would be for me to study programming and computer science.

When I became 24 years old, it occurred to me that I almost certainly don't want to do programming or software development for a living anymore, which was surprising for me since I seemed to check off all the boxes for being good at it (at least superficially). I've written a separate file here about why I think I'm better off trying to become an actuary instead.

If anybody has read this far, thanks for reading all this. I'm predicting that most people won't find my story to be very interesting, but I'm writing it anyway for my own reference and analysis as an autobiographical essay.

4.1 Object-Oriented Programming Is Overrated

Read More:

- Out of the Tar Pit
- Object-Oriented Programming is Bad
- Object-Oriented Programming is Embarrassing
- Object-Oriented Programming is Garbage: 3800 SLOC example
- Object-Oriented Programming is Good*
- Semantic Compression Programming
- Brian Will's Github

- Critique of Code Aesthetic's Nesting Code: From C to C++ to Rust

5 Conclusion And Plans For The Future

I wish I had figured out my strengths earlier in life, and I wish I had someone to guide me and sharpen my strengths during my teen years. I had always been hoping that I'd have time to really learn fast and effective programming skills, but familial issues that were caused directly or indirectly by my parents prevented me from exploring them earlier during college.

I'm not going to completely give up on programming, but I definitely don't want to get career where it's the main focus. There are some linguistics-related programs that I really want to create because I haven't been able to find any other software that can do the things that I'm thinking of. If I complete them, they'll be the first major programming projects that I've ever planned out and created all by myself. Sometime, I may post another file on my blog about all the programming project ideas that I have. Some of them will be ideas that I no longer have plans for, and some of them will be ideas that I might be able to do myself, perhaps with some help.

Possibly, the best way for me to ever become a full-time software developer (especially for a company) may be for someone else who also uses Doom Emacs as their integrated development environment were to show me how to work as efficiently as possible for the Emacs modes, keybindings, programming language, APIs, and such. One of the problems is that there are ways to fix things that irritate me regarding IDEs, but they'd take too much time and expertise to figure out how to resolve, compared to when someone else already has the answers on how to fix and resolve those problems instantly and immediately.