Daniel Diseroad

CMSC 331 TuTh 2:30 PM

Homework 1

1. Paul Graham made some predictions (now over 12 years old) about what popular programming languages will be like in 2103. Of these, describe the ones you find most and least believable in a few paragraphs. (20 points)

   Graham states that in the future, the advanced hardware in computers will allow a shocking amount of inefficiencies in programming languages even compared to some of the wasteful new languages we have now. I find this idea extremely believable, when you take into account Graham's description of how efficient programs had to be in the past just to have them run. Even in my limited experience as a programmer, I can say with certainty that in most cases efficiency isn't the number one thing on a coder's mind. Programmers just want a program to run properly, efficiency will become less and less important as time goes on and technology evolves.

   I find Graham's ideas about strings and their importance the least believable. Graham believes that "getting rid of strings is something we could bear to think about." With Graham's ideas about 'good waste,' I'm surprised that he believes this. Although strings are wasteful in comparison to using arrays of characters, the real world time that strings save are something that will keep them in use. Especially with the libraries and tools that most modern languages have for strings, I don't believe that programmers would get rid of strings so quickly.

   Graham's ideas about parallelism and, more accurately, the waste of its potential are something that I do agree on. I don't see parallel computation as an idea that will be used by the average programmer. As Graham states earlier in this essay, future programmers will be less concerned with their waste of computer resources. Parallelism is just an idea that will fall under this waste. I do believe that parallelism may become more common as programming evolves in the future, but I also agree with Graham and his ideas about the use of parallelism in popular programming languages in the future.

2. We usually think of programming languages as allowing us to specify procedures that can be carried out by computers. Describe the concept of a non-procedural programming language and give at least two examples of languages generally considered to be non-procedural. (10 points)

   A non-procedural programming language is one that rather than having a programmer specify how the program should work and directions to achieve its task, it requires the programmer to describe the problem without defining a solution. Both SQL and Prolog are generally considered to be non-procedural programming languages.

3. Describe the most important contributions that the programming language Algol made in a paragraph or two. (20 points)

   Algol introduced several different features as well as concepts that were adapted to other programming languages. One of Algol's innovative design criteria, orthogonality, allowed for several innovations in itself. Its relatively small amount of basic constructs and rules for combining those constructs allowed for a decent amount of freedom within the language. One result of this orthogonality is the concept of user-defined data types that allowed users to design data abstractions that could fit their specific problem. Algol also introduced dynamic arrays, that for which the size of the array is set at the time memory is allocated. In addition, Algol introduced the concept of reserved words, that in which the symbols used for keywords are not allowed to be used as identifiers by the user.

4. Your roommate claims that it's no longer necessary for computer science students to learn low level languages like assembly or even C because we have much better high-level languages like Java and Python that allow programmers to be more productive. Explain why you agree or disagree with her. (20 points)

   I agree to some extent with my roommate. Although assembly and C are somewhat outdated compared to the languages we use today, I think there is much we can learn from these older languages. Many of the languages we have today have specific libraries and built-in functions for code that you would have had to write yourself in the past. I believe that this challenge helps people to become better programmers by causing them to think outside of the box and make their own solutions to these problems. Learning these low level languages does give you an appreciation for how far we've come in the computer science field.

   Even so, I agree with my roommate in that we have much better languages now, not many employers will fault a programmer for not having extensive knowledge of assembly and C. Some people believe that these low level languages are necessary to learn to "see where we came from" in terms of how programming originated. Although I appreciate the sentiment, I don't think that these languages are necessary to learn in order for a computer science student to become a successful programmer in our modern workforce.

5. Another classmate believes that the current collection of high-level programming languages (C, C++, Java, C#, Python, and Perl, among others) are all anyone will every really want. She doesn't expect significant new ones to be developed and to become widely used. In a paragraph or two, explain why you agree or disagree with her. (20 points)

   I disagree, computer science has already shown rapid evolution of new programming languages and what languages are popular at a given time. Many programming language developers seek to make the perfect programming language, or at least one that offers improvements over the current languages. There will always be new evolutions in hardware that need new languages to take advantage of, that trend will continue. Graham

speaks about the prospect of a hundred-year language and how it is possible that programming language could be developed today. Based on the history of programming languages, I don't believe that is true. Computer science has often made progress step by step, not by huge leaps. This trend will most likely continue, creating many new programming languages as a result. Just as I had never heard of Algol, Colbol, or APL, I can believe that one day maybe the likes of C++, Java, and Python will be unknown to new programmers. The field of computer science is not a stagnant one, innovations will continue to exist and change the field along with the programming languages we see as common.