121/125 Name: Adam Stammer

Exam 1, CS-415: Principles of Programming Languages

Spring 2020, Iyengar

List and describe at least three differences between natural languages and programming languages. (10)

Programming languages tend to be far less ambiguous than than natural languages, making them easier for computers to interpret.

Prog. Lang. tend to be much simpler than natural languages, making them much easier to learn

Prog. Lang. tend to express structural ideas much more acurately than natural languages are capable of.

(-1)

What general properties are desirable in any programming language? (10)

Simple → simple is easy to understand, read, and write

Ease of Use, Learning, and Teaching → A language must be accessible to be used and the more it's used the more it progresses.

Unambiguous → don't leave things to guessing or misinterpretation, it only complicates things

Efficient → cheap is good. Make it fast to code and fast to run.

Unrestricted →

What hardware capability that first appeared in IBM 704 strongly affected the evolution of programming languages. Explain why. (10)

It introduced floating-point calculations built directly into hardware. Not only did this make floating point calculations much faster and more efficient, but it also simplified the code required to do floating point calculations. This set a precedence for future computing.

index (-1)

Define Automation principle. Give an example of the use of Automation principle in Fortran. (10)

Automate mechanical, tedious, or error prone activities

The original punch cards had to be in order but it was easy to mix them up or drop a whole stack, so card sorters were invented to automate this

(-2)

Define Security principle. Give an example in Fortran that violates this principle. (10)

No program that violates the definition of a language should be able to escape detection.

Typos in variable names would be treated as new variables not caught as an error and could easily lead to massive frustration.

efine Abstraction principle. Give an example of the use of this principle in Fortran. (10)

Avoid requiring something to be stated more than once.
Factor out the reocurring pattern.

Subroutines allow a piece of code to written only once but used over and over, thus avoiding the repeat of the same code.

What is the Zero-1-infinity principle and why is it important? (5)

The only reasonable numbers are zero, one, and infinity.
If you are going to limit something, limit it to one, or don't allow it at all.
It is hard to remember multiple abstract quantities, like say the functional variable name length limit, so, it's best to add such a limit. (or the for loop statement limit)

What is the Structure principle? Describe its importance. (10)

The static structure of the program should correspond in a simple way to the dynamic structure of the corresponding computations.
This is vital to avoiding spaghetti code. A whole bunch of "IF... GOTO..." statements is complicated to read which slows developement down.

What were the primary purpose of Algol? (10)

To be used by scientists as a very efficient, machine code based language, that solved many of the spaghetti code based problems of other languages. It was very aimed at research and publication.

Why did it fail commercially? (10)

The lack of built in I/O made the language far less portable and more difficult to learn. I/O came from machine specific libraries that were different for each machine and had to be developed for each machine. Without portability and ease of use, it's no wonder it didn't catch on.

Describe the contribution by Algol (with an example) pertaining to:

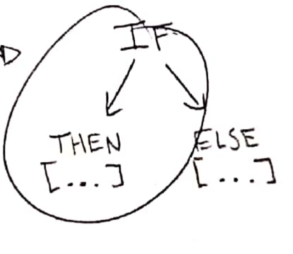
Basic Syntax (5)

Keywords → (begin, end) that allowed free formatting
but still no statement terminator

Syntactic Notation for Grammars → E :- C

Program Structure (5)

Hierarchical Structure →



Concise blocking built into the language made it much easier to read and follow code

(r1

List at least 3 characteristics of First Generation Languages. (10)

Static Memory Allocation → maximum possible memory space was used regardless of the actual requirements

Weak Typing → type checking was rare or entirely absent which makes type errors less likely to be caught / during compilation

Machine Orieated Syntax → Languages were efficient because they were designed to the machine, but not user friendly making developement more difficult

List at least 3 characteristics of Second Generation Languages. (10)

Dynamic Memory Allocation → only used the memory it needed at that time, allocating and deallocating as needed

No Implicit Delaration → a variable name typo was much more likely to be caught by a compiler /

Static Scoping → prior to this, subroutines could only share information via parameters. This made data sharing throughout a program easier and more simple.