**Programming languages ✓**

- What does 'paradigm' mean?

A way of doing something, a style, a mindset.

- What are the different programming paradigms?

Functional, Object Oriented, Logical, Imperative/Procedural

- What does 'syntax' mean?

The arrangement of words, a set of rules for how to write

- What does 'semantics' mean?

The meaning of those words or statements.

- Why are there so many programming languages?

There are many programming languages because there is no one best programming language for anything. Many languages have strengths in one area, and weaknesses in area. Languages also evolve and add/remove features, which would create new languages just by evolving.

- How to classify programming languages?

Classify them by generation, high level/low level, paradigm, compiled vs interpreted, scripting language vs system languages.

- What are the programming language generations?

The first generation is the lowest level machine language, just 0’s and 1’s

The second generation is assembly language

The Third generation is the highest level, and are the compiled/interpreted languages we use today

- What is a compiler?

A program that turns source code into machine language that a computer processor can read and then execute.

- What is an interpreter?

A program that reads source code line by line and is then executed line by line without first compiling it into machine language.

- How an interpreter is different from a compiler?

An interpreter reads a program line by line and executes the program directly, while a compiler first turns the entire program into machine language, and then the machine language code must be executed.

- What is bytecode?

Bytecode in java is the compiled version of the java source code that is read and executed by the Java Virtual Machine.

- What is your favorite programming language?

Python

- What are the indispensable properties of programming languages?

The only real indispensable property of a programming language is that it works as expected, and in a reasonable amount of time.

- What is a 'good' programming language?

Good syntax, fast execution, good memory management, good exception handling, good data types and semantics

- What makes a programming language popular?

The most popular programming languages, C, C++, Python, Java, are all universal languages. They can be used in almost any industry, and that helps them achieve their popularity. Python and Java are also languages that beginners to programming use for learning.

- Show on an example how SIMD and MIMD work.

If you have two arrays [0,1,2] and [3,4,5] and you want to add each element that’s in the same position and add the result into a third array, this would be an example of SIMD. Single Instruction Multiple Data, it is a single instruction; addition; run on multiple data; the array elements. If you want to add the first elements, subtract the 2nd elements, and multiply the 3rd elements, then it would be MIMD; multiple instruction multiple data.

- What is SQL?

Structured Query Language. A specialized language for databases

- What is Prolog?

A general-purpose logical programming language. Used a lot for A.I

- How do you learn a language?

You learn the syntax of the language, and then you simply learn it by using it. The same way you get to Carnegie Hall, you practice.

- Write a small prolog program based on the example of the slides.

P(Edward VII, George V).

P(Victoria, Edward VII).

P(Alexandra, George V).

P(George VI, Elizabeth II).

P(George V, George VI).

P(Elizabeth II, Charles)

P(Charles,William)

P(William,George)

P(Kate,George)

P(William,Charlotte)

P(Kate,Charlotte)

G(x,y):-P(x,z),P(z,y).

- How to classify languages?

You can classify them by generation, compiled/interpreted, high level/low level, and by paradigm.

- Is the paradigm classification up-to-date?

No, more and more languages today are multi-paradigm.

- What did you learn about language design choices in this chapter of the course?

I learned how programming languages are classified. The schemas for designing programming languages (Which paradigm to base it on). I learned how important good syntax can be, and the history of programming languages and their designs.

- Example of a logical, functional, imperative, and Object Oriented Programming language?

Logical – Prolog, Functional – LISP, Procedural/Imperative – FORTRAN, OOP - Java

**Recursion**

- Where is recursion used?

Recursion is used whenever a function is defined in terms of itself (calls itself) It is used a lot in functional programming languages

- What does iteration mean?

When a set of instructions is repeated a specified number of times or until a condition is met

- What is an explicit function definition?

When a function consists of an expression that indicates what the output is for each element X in the given domain from previously defined functions

- What is a recursive function definition?

When a function F indicates how f(x) is obtained from previous defined functions and values of f(x) for smaller arguments

- What is a well-formed recursive function definition?

When it has a basis case (The output for the smallest argument), and a general case which is calling the function itself but for called for smaller arguments

- What is a well-defined recursive function definition?

If F(n) can be computed for all n for which the function is defined. (For example in Java, Factorial is not well defined because it cannot compute above 20!)

- True/False: A recursive function defined with a well-formed definition always terminates.

True

- True/False: A well-defined recursive function always terminates.

False

- Computer fact(4).

Fact(4) = fact(3) \* 4 = fact(2) \* 3 = fact(1) \* 2 = fact(0) \* 1 = 24

- Computer fib(5).

Fib(5) = fib(4) + fib(3) = ….. = 5

- Why is the computation of fib(n) exponential?

The tree grows exponentially because you are repeating a large number of computations you have already done.

- What is the Ackermann function?

One of the simpliest and oldest examples of a total computable function that is not primitive recursive.

- What is the McCarthy 91 function?

A recursive function that always returns 91. If n is greater than 100, return n – 10, else if N is less than or equal to 100, then return M(M(N+11))

- What is the innermost/outermost/simultaneous evaluation of recursive function?

Ok

- True/False: Innermost evaluation always terminates.

False, but more efficient

- True/False: Outermost evaluation always terminates.

True

- True/False: Innermost evaluation is more efficient than outermost evaluation.

True

- Describe the Merge sort algorithm.

Split the array into halves until you have arrays of size 1 (which are defined as sorted), then merge them back piece by piece sorted until you have a sorted array

- Why is the number of moves in the Tower of Hanoi problem exponential?

It is 2^n – 1 moves to complete 3-peg tower of hanoi. To complete tower of hanoi takes as many moves as to put all disks by 1 on the opposite peg, then + 1 to move the last peg, and so on and so forth recursively. Making it exponential

- What does undecidable mean?

It means it cannot be determined if a program will terminate or not.

- What is the Halting problem?

The Turing Halting problem is whether or not a function can be made, that accepts a program as an argument, and tells whether or not the program will terminate.

- True/False: Recusion is always less efficient than iteration.

False, tail recursion is just as efficient

- What is tail recursion?

A special form of recursion that is much more efficient than normal recursion. Every tail recursive function can be made iterative, and it will be just as efficient. A recursive function is tail recursive if it does not contain any other function other than itself (No multiplication,etc). If C(x) then A(x), else F(p(x)) Factorial is not T-R (multiplication), but GCD is

- How to transform a tail recursive function definition into an iteration? Refractoring

Y = x;

While (not c(y) )

{ y = p(y) }

Return a(Y)

- When should one use recursion?

When one cares more about elegance than performance, and memory is not an issue. Recursion is usually more readable than iterative definitions

- You must be able to write recursive definitions of functions.

Ok. Tailfact(n, 1) = fact(n)

Tailfact(n,x) = if n = 0 then x, else tailfact(n-1, n\*x)

Y = 1;

While ( n > 0 ) {

Y = n \* y

N = n – 1 }

Return y;

- What did you learn about language design choices in this chapter?

One of the biggest takeaways for me was innermost vs outermost evaluation, it was something I had never considered before. I learned whether or not to include tail recursion support in a compiler to automatically generate an iterative version. I learned more about the structure of recursion design (well formed vs well defined).

**Java**

* .equals and ==
  + .equals compares the VALUES of both objects
  + == checks the REFERENCES and sees if they point to the same thing
* Clone
  + Clones an object, either by making It refer to the same object, or by having it contain the same value.
* CompareTo
  + Returns 0 if equal, returns -1 if less, returns 1 if greater. On strings it is based on string length (lexicographically)
* Implementing functions such as fact and fib

Can do

**Older languages**

* Read the presentations of your classmates. Identify some important features of the presented languages. Alright

**Memory Management**

- What is memory management?

The process of binding values to memory, taking into account both the static and dynamic characteristics of these values

- What are the static memory / run-time stack / heap?

Static memory are values that are known to the compiler BEFORE run time and remain constant through out the execution. The run time stack is the center of control for evaluating functions/methods, their local variables, and parameter-argument linkage. The heap contains all other values that are dynamically made during run-time. It is the least structured

- What is parameter passing?

Parameter passing by value gives the function just the values, meaning it cannot change the variables that it gets (Also called Pass by copy). Pass by reference gives it the exact address location of the variable it gets, meaning it can make changes to it

- What are sigma, gamma and mu and how do they permit to understand memory management? How do declarations, assignments, allocations and deallocations affect sigma, gamma and mu?

The environment of a variable is given by Gamma, the memory location is given by Mu, and the state of the variable is given by Sigma. Sigma is given by Gamma x Mu.

- What data structures to use to represent the memory?

A stack

- What is an argument / a parameter?

A variable passed to a function

- What is static / dynamic scoping?

The scope of a variable determines where a variable can be used.

Static scoping means the scope of each variable is determined by the static structure (using static links) Dynamic Scoping means the scope of each variable is determined by the sequence of method’s calls that occurs at run time (using dynamic links)

- How are arrays saved in memory?

They are ordered sequences of values with the same type, it just means they are directly adjacent to one another in memory

- You must be able to trace the memory management of Java and other programs.

Ok, remember the order (Static, main, runtime) and remember the arrows and boxes for static/dynamic at the beginning of every new section (main, new function,etc)

- How is garbage collection done in Java?

Java uses automatic garbage collection with a low priority process called a thread. System.gc()

- What is null in Java?

An unset reference, a variable does not refer to anything

- What is a wrapper class in Java?

A wrapper class is a class that encapsulates the eight primitive data types, hiding them, so that they can turn data types into objects with their methods.

- What is an immutable / mutable type in Java?

An immutable type means that once an object of that type is made, it’s value cannot be changed. A mutable type means that once an object of that type is made, it’s value can still be modified.

- Exercise: Considering the K package what are the values of a, b, h and i if:

a) parameter passing is by value for x and by reference for y in A, and by reference in B

Done on looseleaf

b) parameter passing is by reference in A and B

Done on looseleaf

- What is garbage collection?

The process of freeing up blocks of memory that are wasted by variables which are no longer needed.

- Explain the 2 garbage collection algorithms: reference counting and mark-sweep, their features, advantages and drawbacks.

Reference counting has object nodes with an integer field that keeps track to what is pointing to it, if nothing is pointing to a node, then it can be freed up. It’s weaknesses however is that it cannot detect isolated field of nodes that only point to each other and nohing else.

- You must be able to trace the garbage collection algorithms.

Mark Sweep also has nodes but with a mark bit field that is seto 0 by default, in the mark phase we set reachable object node’s mark bits to (1), then the sweep phase occurs that deletes all unreachable objects. The disadvantage of it is that normal program execution is suspended while it runs. But it will work on isolated circular cycles of nodes.

Ok

- What did you learn about language design choices in this chapter?

How memory is actually managed behind the scenes. What exactly happens during run-time and execution of a program. How garbage collection works and the different methods for implementing garbage collection.