Andrew Morris

2-1-16

CSCI 330

Chapter 1 questions

1. It enables the programmer to analyze languages from a lower level. It allows us to break up the compiler/interpreter into separate pieces, and gain insight into these processes.
2. It increases our capacity to use different constructs in writing programs, enables us to choose languages for projects more intelligently, and makes learning new languages easier.
3. Fortran
4. COBOL
5. LISP
6. UNIX
7. It increases the complexity of the features of the program i.e. having too many ways to increment a variable increases the complexity of incrementing a variable.
8. Because overloading an operator takes sensibility. If a operator is not overloaded sensibly, then the reader of the program could easily misinterpret the intentions of the author.
9. The availability to return an array vs. returning a struct. An array can be of a type, except void and a function, which cannot be returned. However, a Struct CAN be returned, which can contain a number of different types. This exception is an example of the lack of orthogonality in C.
10. ALGOL 68
11. If-Then
12. The ability to call functions.
13. A program is reliable if it performs to its specifications under all conditions.
14. If a program does not perform type checking on subprograms, then a unexpected type of variable can be passed. This will distort the results of the program.
15. Aliasing is having two of more distinct names that can be used to access the same memory cell.
16. Exception Handling allows a computer to intercept run time errors without the program exiting.
17. Readability is important to writability because if a prom tram is hard to read, then it is hard to write.
18. It depends on how reliable a program is supposed to be. If a program is to be compiled several times then the compiler does not need to perform much optimization on the code. If the code is being used for a long term system, then the extra cost for a compiler might be warranted.
19. Prevalent Computer Architecture.
20. Imperative Languages
21. Top-down design and stepwise refinement
22. Data Abstraction, Inheritance, and Dynamic Method binding
23. Smalltalk
24. Readability and Simplicity
25. Compiled, Interpreted, or a hybrid of both (ex. JIT)
26. Compiled languages provide faster program execution.
27. It serves as the database for the compilation process
28. It links the systems programs to the users programs.
29. The Von Neumann bottleneck is what limits the speed of a computer, and is what is generally accepted as the computer’s speed.
30. The program is completely portable from system to system as long as it has an interpreter for that language.

3. If there was only a single language for all programming domains, then there would be a high domain of knowledge for that particular language. Also, this would increase the maintainability of code, and the reuse of compilation technologies. I would also argue that portability would not be an issue because hardware would tend to be designed to leverage the language.

4. I can argue that based solely on the variety of needs for different types of programs, that different languages are needed. For Example, creating a GUI based program in Java is fairly easy, where implementing a Gui based program in C would be VERY challenging.

5. Another criterion that a language can be judged is if the language is strongly, or loosely, typed. In a strongly typed language, each variable must be explicitly assigned a type. In a loosely typed language, the type of a variable is not explicitly stated. For example, in C, which is strongly typed. Creating a float variable looks like this “float x”. In a loosely typed language, like python, declaring a float is done like this “x = 2.0”. As you can see, the type in python is not explicitly declared.

7. A argument against having just the right brace is that it can be difficult to determine which group statement is ended by each right brace. A argument for is that it is simple and straightforward how to end a statement because the programmer does not have to individually label each closing statement.

8. Pros and Cons of distinguishing upper and lower case letters in variable names.

Pros: Able to use a greater number of variable names. Also, there are ways to make a variable name appear to be more readable. For Example, if we nee a counter for the number of books we can use “numberOfBooks” rather than “numberofbooks”.

Cons: The issue with distinguishing between upper and lower case is that it can be difficult to remember the exact syntactic specifics of each variable. This can lead to injected bugs by having misspelled variable names that are off by case.

10. Just because hardware is becoming more inexpensive, this does not mean that we should waste resources. Also, all computers that run a program benefit from the efficiency of a program. In design, we do not know all of the speeds of computers that will run the program, therefore we should make them as efficient as possible in order to increase the range of computers that can use the program.

15. Using type declaration for simple variables affects the Overall Simplicity of the code. In that if a simple counter is used, in order for the maintainer to edit the code, they must also search for the declaration of that variable.

18. Having multiple line comments is beneficial when we want to comment out a range of code for design/debugging. Also, multiple line comments allow for larger, and more obvious summary comments. Mulitple line comments do suffer though, because if there is a very large comment block, it can be difficult to find the beginning and ending of the comment block.

Vice versa, Single line comments are difficult to use when producing a large comment block. If single line comments are used, the comment syntax can become a hindrance to the readability of the comments. The handy thing about single line comments is that we can have comments after a line of code to produce a simple explanation. Usually these single line comments are easier to type on account of their identifiers.

PART II:

1. Orthogonality means that a relatively small set of primitive types can be combined in a relatively small number of ways to build the control and data structures of a language. An orthogonal language is the VAX language. A non-orthogonal language is the IBM mainframe instruction set.

An example of orthogonality can be found in the VAX language. In VAX, adding two integers can be done by “ADDL operand\_1 operand\_2”. In this statement, either register a register, or a memory location can be used as the operand.

An example of non-orthogonality is the IBM mainframe, in which the instruction is different from when the operand is in a register, or if the operand is in memory.

1. Programming Domains
   1. Scientific

PROGRAM MAIN

INTEGER N, X

EXTERNAL SUB1

COMMON /GLOBALS/ N

X = 0

PRINT \*, 'Enter number of repeats'

READ (\*,\*) N

CALL SUB1(X,SUB1)

END

SUBROUTINE SUB1(X,DUMSUB)

INTEGER N, X

EXTERNAL DUMSUB

COMMON /GLOBALS/ N

IF(X .LT. N)THEN

X = X + 1

PRINT \*, 'x = ', X

CALL DUMSUB(X,DUMSUB)

END IF

END

* 1. Business

SD WorkFile.

01 WorkRec.

02 WStudentId PIC 9(7).

02 FILLER PIC X(23).

PROCEDURE DIVISION.

Begin.

SORT WorkFile ON ASCENDING KEY WStudentId

INPUT PROCEDURE IS GetStudentDetails

GIVING StudentFile.

STOP RUN.

GetStudentDetails.

DISPLAY "Enter student details using template below."

DISPLAY "Enter no data to end.".

DISPLAY "Enter - StudId, Surname, Initials, YOB, MOB, DOB, Course, Gender"

DISPLAY "NNNNNNNSSSSSSSSIIYYYYMMDDCCCCG"

ACCEPT WorkRec.

PERFORM UNTIL WorkRec = SPACES

RELEASE WorkRec

ACCEPT WorkRec

END-PERFORM.

* 1. Artificial Intelligence

(defun factorial (N)

"Compute the factorial of N."

(if (= N 1)

1

(\* N (factorial (- N 1)))))

* 1. Systems Programming

…

printf("This process' ID is %d\n", PID);

printf("Please specify the number of alarms you would like to set?\n :");

scanf("%d", &numAlarms);

currTime = time(NULL);

printf("Current Time:%d \n",currTime);

printf("Setting alarms...");

/\* putting alarm times in the queue \*/

for (i = 0; i < numAlarms; i++) {

randomNum = rand()% (MAX\_TIME) + i;

currTime = time(NULL);

queue[allSigs] = randomNum + currTime;

printf("(%d, %d)\n",i, (currTime + randomNum) );

++allSigs; /\* Incrementing the amount of signals \*/

}

…

* 1. Web software

<!DOCTYPE html>

<html>

<body>

<?php

$x = 5; // global scope

function myTest() {

     // using x inside this function will generate an error

     echo "<p>Variable x inside function is: $x</p>";

}

myTest();

echo "<p>Variable x outside function is: $x</p>";

?>

</body>

</html>

1. A compiler takes source code in a language, and through a series of analytic processes, generates a binary executable that is specific to the computer’s hardware on which it was compiled.

An interpreter does not have to modify the file into machine code to run on a particular machine. As long as the machine has the interpreter installed, then it can execute the code.

Compilers and interpreters are similar in that they both use source code(Interpreted or compiled) to generate a program on a computer.

They differ greatly in the fact that purely interpreted languages cannot undergo optimization because there is no compilation process. Also, machine code for a computer is very unportable, and source code for a interpreted language is very portable; as long as the correct interpreter is installed.

1. The source code is compiled via a C++ compiler. The source code goes through lexical analysis where the lexical units are formed. The syntax analyzer then constructs the parse trees needed, if any, to analyze the syntax of the program. Then the intermediate code generator produces a assembly-like language for semantic analysis. Then the program is optimized, but for ‘hello world’ there is not likely any difference the optimization could make. The optimized code is now generated into machine instructions. The compiler will link the output systems programs needed for displaying ‘Hello World’. The executable is now created. When the user runs the program, it is loaded into memory, and calls the necessary systems programs to display ‘Hello World’.