Programming Language Paradigms

COMP360

“When a programming language is created that allows programmers to program in simple English, it will be discovered that programmers cannot speak simple English.”

Different Approaches

- Not every language is like Java with slightly different syntax
- Some programming languages specify how the computer is to solve a problem. Others specify only what the solution should be
- Some languages are a simple extension of machine language while other are based on a more logical view

Language Paradigms

- Assembler Languages
- Procedural Languages
- Object-Oriented Languages
- Functional Languages
- Rule-based Languages
- Declarative Languages

Assembler Languages

- Assembler language specifies the exact machine language instructions to be executed
- Syntax is usually very simple and not a context free language
- Assembler is architecture specific

Assembler Extensions

- Some assembler languages have one mnemonic for each machine language instruction
- Other assemblers produce different machine language for a mnemonic based on the addressing, data size and direction of data movement
- Many assemblers support macros

Writing an Assembler

- The first COMP360 assignment has been posted on Blackboard
- Write an assembler that reads the source code of a simple assembly program and displays the machine language for that program
- It is due by noon on Friday, January 27

Example Output of Assembler

```
address  machine     source
         language
                     * Example program for COMP360
                     *
       0  58000025           la   R8,addr
       4  4180002e           ld   R1, stuff[R8]
       8  42800025           ld   R2, addr[R8]
       c       123   more    add  R1, R2, R3
       e      2324           mult R3, R2, R4
      10  b400000c           jn   R4, more
      14      1415           sub  R4, R1, R5
      16  a300001c           jz   R3, nodiv
      1a      3536           div  R5, R3, R6
      1c  66000025   nodiv   st   R6, addr
```

Symbol Table

- Create a symbol table that keeps track of all labels and their addresses
- Put the mnemonics in the symbol table including
  - opcode
  - format
  - length
- Put register names (i.e. R1, R2, etc.) in the symbol table

General Outline, Pass 1

- Initialize the current address to zero
- Read a line of assembler
- Split the line into tokens
- If label, put label and current address in symbol table
- Get mnemonic and find in symbol table
- Add instruction length to current address
- repeat until end of file

General Outline, Pass 2

- Initialize the current address to zero
- Read a line of assembler
- Split the line into tokens
- Get mnemonic information from symbol table
- Get additional fields and create the machine language
- Display the results

Why is Dr. Williams so cruel?

- An assembler is the simplest language translation system we will study
- You will get experience with symbol tables and translation
- If you can convert from assembler to machine language, then you know you can create machine language if your program creates assembler

Procedural Languages

- The program concentrates on the flow of control
- The most common type of programming language
- Early programming languages followed this paradigm
- The programmer specifies a step by step procedure for solving a problem
- Examples: C, Cobol, Fortran, Pascal

Click the one true statement

A. All C++ programs are valid C programs
B. All C programs are valid C++ programs
C. C++ and C are completely different
D. C++ and C are the same

Procedural or Imperative Languages

- The model of execution is the changing of the machine state, the values in memory
- Programs are a sequence of statements that modify the values of variables to arrive at a solution

GoTo

- In addition to the familiar block structure constructs such while loops, ifs and for loops, early programming languages usually had a **`goto`** statement
- This allowed you to jump to any labeled statement
- The idea is a simple extension of the assembler jump
- Using **`goto`** statements can lead to “Spaghetti code” whose logic can be very hard to understand

Structured and Unstructured Code

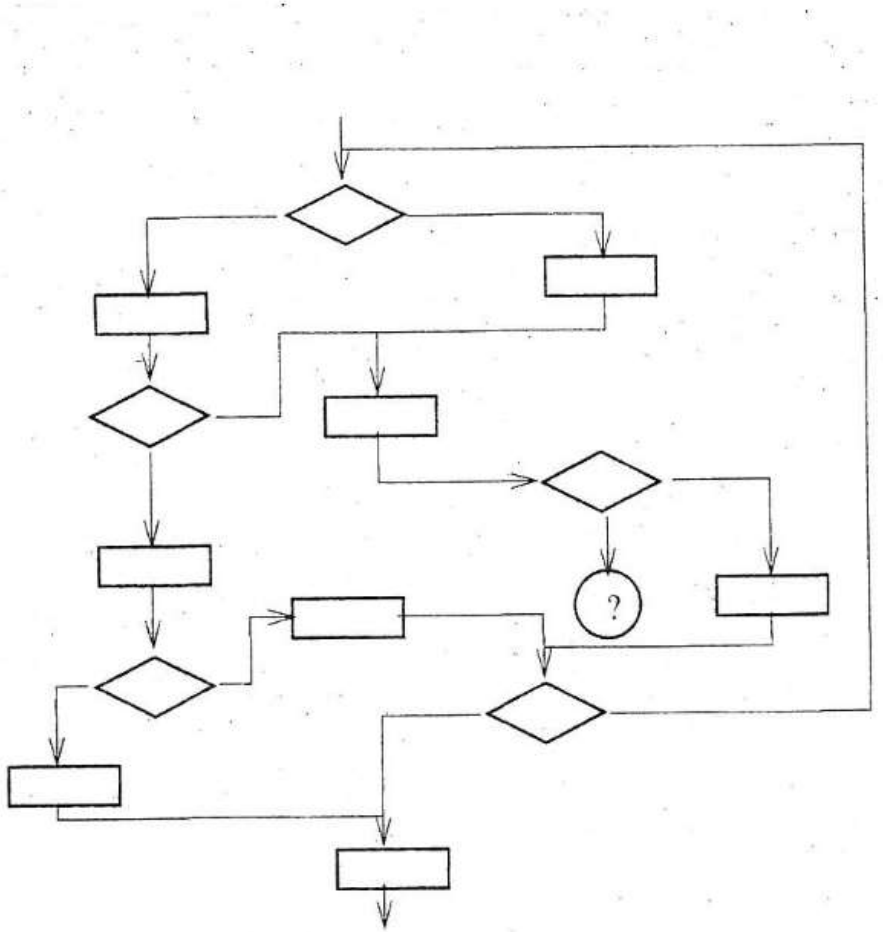
```
while dog > 0 {
    if cat == dog
        goat = 1
    else
        goat = 2
    print goat
}
```

```
5  if dog <= 0 goto 7
   if cat == dog goto 4
   goat = 2
   goto 3
4  goat = 1
3  print goat
   goto 5
7
```

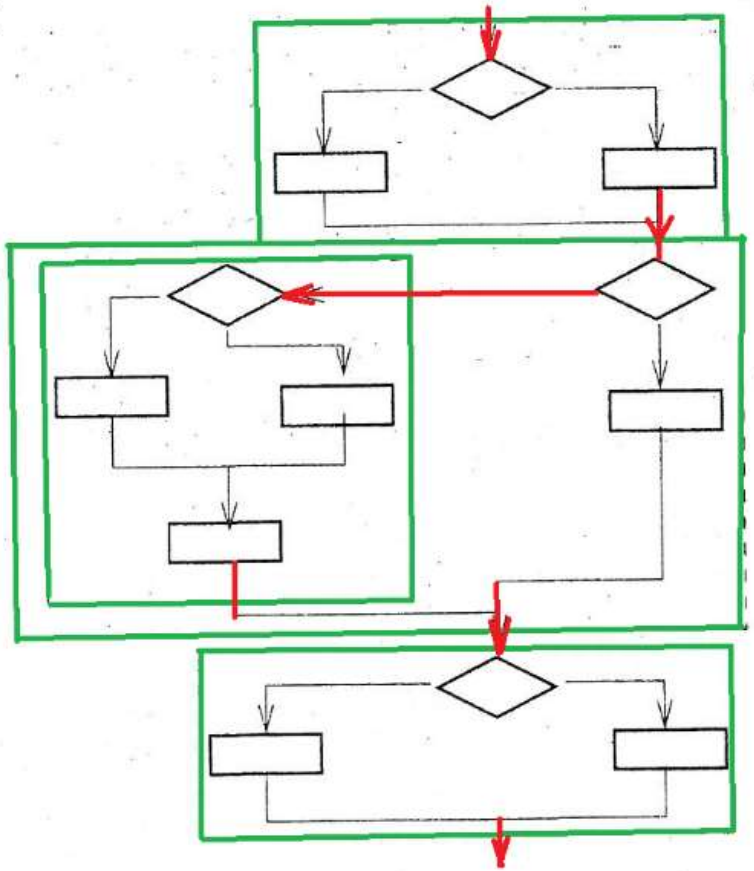
“For a number of years I have been familiar with the observation that the quality of programmers is a decreasing function of the density of go to statements in the programs they produce.”

Edsger W. Dijkstra

Blocks Have One Entrance and Exit



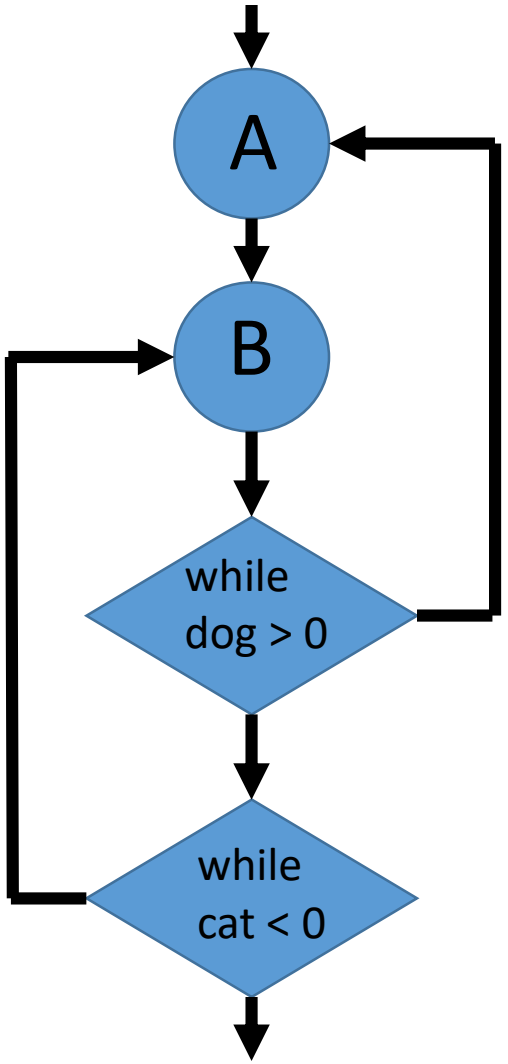
(a) Spaghetti code



(b) Structured code

from Terrence Pratt, “Programming Languages”

Write this is in Java



Algorithm Restrictions

- There are some algorithms that are not easily written in block structured languages like Java or C++
- Block structures prevent entering or leaving a loop in the middle
- Almost all languages now are block structured

goto in Java

A. Java supports goto
B. Java does not allow goto
C. goto is a keyword in Java, but it doesn’t work
D. None of the above

Object-Oriented Languages

- Object-Oriented Languages are generally an extension of procedural languages
- OO connects the possible actions with the data. The programmer defines the data and the possible functions on the data.

OO continued

- OO provides:
  - Inheritance
  - Encapsulation
  - Polymorphism
- Examples: C++, Java, C#, Smalltalk

Functional or Applicative Languages

- Based on the idea of mathematical functions, but not limited to math
- Functional programs usually use recursion instead of iteration
- Examples: LISP, ML and Haskell

Java and C++ support

A. Inheritance
B. Encapsulation
C. Polymorphism
D. All of the above
E. Only some of the above

Functional Languages

- Look at the function the program represents rather than just the state changes
- What is the function that must be applied to the input to get the desired output

$$function_n(\ldots function_2(function_1(data)) \ldots)$$

```
-- capitalize first letter of each word in a sentence
capTitle (' ':goat:cat) = ' ' : toUpper goat : capTitle cat
capTitle (dog:cat) = dog : capTitle cat
```

Rule-based or Logic Languages

- The programmer establishes a set of relationships between the data
- The program seeks a solution that meets the specified relationships
- The language is designed to execute some kind of logic “engine”
- Example: Prolog

Rule Based Execution

- Rule based languages execute by checking for the presence of enabling conditions and, when present, executing an appropriate action

$$condition_1 \rightarrow action_1$$
$$condition_2 \rightarrow action_2$$
$$\ldots$$
$$condition_n \rightarrow action_n$$

from Terrence Pratt, “Programming Languages”

Declarative Languages

- When using a declarative language, the programmer does not specify how the results are to be created
- The programmer specifies exactly what the results are to be
- Examples: SQL and Report Generating Languages

Structured Query Language

- Structured Query Language (SQL) is used for database access and management

```
SELECT Book.title AS Title, count(*) AS Authors FROM
Book JOIN Book_author ON Book.isbn =
Book_author.isbn GROUP BY Book.title;
```

- Example output might resemble

```
Title                     Authors
------------------------- -------
SQL Examples and Guide    4
The Joy of SQL            1
An Introduction to SQL    2
Pitfalls of SQL           1
```

Near Term Schedule

Martin Luther King Day holiday *(no classes)*	Wednesday, January 18 Programming paradigms	Friday, January 20 Programming in assembler
Monday, January 23 Programming in assembler	Wednesday, January 25 Computing theory read 2.4	Friday, January 27 Language translation read section 2.1

Writing an Assembler

- The first COMP360 assignment has been posted on Blackboard
- Write an assembler that reads the source code of a simple assembly program and displays the machine language for that program
- It is due by noon on Friday, January 27