

APM 153 LECTURE TWENTY EIGHT Programming Languages, Assembly Language, High Level Languages, BASIC, Programmable Calculators

Programming Languages

(1) The development of the first electronic, digital computers was closely followed by the development of computer languages that could be used to program those computers.

(2) Before the development of computer languages, each new calculation was set up by physically rewiring the circuits in the computer.

(3) Computer languages made rearranging circuits unnecessary and allowed for programs to be stored, tested, and re-used. One of the first languages was Assembly Language.

Assembly Language

(4) Assembly language - or simply “assembly” is a human-readable notation for the specific **machine language** that each type of computer uses.

(5) Assembly language makes machine language readable by replacing raw numeric instructions with symbols called **mnemonics**.

(6) For example, a set of instructions in machine language might look like this, ..

10110000 01100001

In assembly language the instructions would be written as,...

mov al, 0x61

(7) These instructions tell the computer to **move** the value **61** into the processor register with the name '**al**'. The mnemonic "mov" is short for "move". This is a typical assembly language statement.

(8) Transforming assembly into machine language is accomplished using a program called an “assembler”, and the reverse by using a program called a “disassembler”.

(9) Because it is so simple, assembly was considered the fastest programming language for a long time. With fast modern processors however, we increasingly use higher level programming languages such as FORTRAN or C++.

(10) Assembly language is still widely used for certain tasks such as writing **operating systems** and **device drivers**. Many **compilers** also render **high-level languages into assembly** first before fully compiling, allowing the assembly code to be viewed for debugging and optimization purposes.

(11) Another common use for assembly language is in the **system BIOS** of a computer to **initialize and test the system hardware** prior to booting the operating system

(12) Finally, assembly language is also often used by hackers to **”reverse engineer” software**. In this case, the software is “disassembled” back into assembly language so that hackers can see how the software operates.

High Level Languages

(13) Even though it was elegant and powerful to use, assembly language was quickly replaced by “high level” languages that were slower, but easier to understand.

(14) The first high-level languages were developed in the 1950s and allowed programs to be written using more understandable mathematical formulas or even English words.

(15) Some of these languages were “interpreted” languages and some were “compiled”.

(16) In an **interpreted language** such as Matlab or VBA, the computer was programmed to directly execute or “interpret” programs one line of code at a time.

(17) In a **compiled language** the entire program is first translated into machine language and then combined with the necessary library routines to compile a working program.

(18) The most popular early high-level languages were FORTRAN, for scientific programming, and COBOL, for business programming.

(19) The number of different languages grew as new ones were created to ease programming different tasks for specific applications. Special languages were created for manipulating ,...

	character strings	SNOBOL
	lists of items	LISP
	vectors or arrays	APL
and	simulations	SIMSCRIPT

(20) A list of the names of every language invented runs to many pages.

Standardization

(21) The development of many different languages and dialects made it difficult to move programs from one computer to a different computer even if they were from the same manufacturer.

(22) This problem led to efforts to create standardized languages such as ALGOL 60 (ALGOrithmic Language 1960), which was developed by an international committee under the initial sponsorship of UNESCO.

(23) ALGOL was widely used, particularly in Europe, but its greatest contribution was to pioneer the “block structure” concept that most languages still use today.

(24) While some languages are verbose, others (such as APL) are remarkably **terse** in how they express a computation. (“terse” means it uses few words)

$\circ\beta(i64) * 3$



An APL program to print the cubes of the first 64 integers

MAIN.

PERFORM INPUT-ADD 10 TIMES.

DIVIDE 10 INTO SUM-VALUE

GIVING AVERAGE-VALUE.

MOVE AVERAGE-VALUE TO EDIT-FIELD.

DISPLAY "THE AVERAGE IS ", EDIT-FIELD.

STOP RUN.

INPUT-ADD.

DISPLAY "TYPE IN AN INTEGER (2 DIGITS)".

ACCEPT INPUT-VALUE FROM SYSIN.

ADD INPUT-VALUE TO SUM-VALUE.



A portion of a simple COBOL program to average 10 numbers

FORTRAN

(25) FORTRAN is the most successful and oldest high-level computer language still in active use. FORTRAN stands for “formula translator” and was one of the first programming languages to allow for-loops.

(26) The development of FORTRAN began in 1954 under John Backus at IBM and the first customer’s program, which stopped with an error because of a missing comma, ran on April 20, 1957.

(27) Like many of the other languages that were being developed in the 1950’s and 1960’s FORTRAN continues to be used and is constantly being improved.

(28) Also like other languages, efforts have been made to standardize the FORTRAN language so that programs written on one type of computer will work on other computers.

(29) When I began learning FORTRAN in 1990, I used F77 which was the standard version promoted in 1977.

(30) Now when I teach APM 360 *Introduction to Computer Programming*, I use F90 which is the 1990 standard. However, the development of FORTRAN continues. The most recent standard is FORTRAN 2000.

(31) Even though FORTRAN is very powerful, there are lots of other programming languages out there in current use such as C, C++, PERL, ADA, Python, and JAVA.

(32) These newer languages are relatively recent developments. There are however, some older programming languages that are still very important such as BASIC.

BASIC

(33) BASIC was created in 1963 by John G. Kemeny and Thomas E. Kurtz of Dartmouth College and stands for **Beginner's All-purpose Symbolic Instruction Code**,

(34) BASIC started as a **compiled** language where the source code you wrote was **converted into machine language**.

(35) In most subsequent versions however, BASIC became an **interpreted language**, where a program interpreted the instructions in your code.

(36) In 1979, IBM began loading Microsoft's version of BASIC into the base memory (ROM) of IBM computers. If your computer did not have a hard drive (and many computers at that time did not), your computer would **boot up** to BASIC.

(37) Current versions of BASIC are more powerful and can be used as either an interpreted language or as a compiled language. Microsoft's Visual Basic and Visual Basic for Applications (VBA) can be compiled.

(38) The version of BASIC in our programmable calculators is also loaded into the ROM

Programmable Calculators

(39) My examples will be specific to the **TI-83 Plus** Series of Calculators, but the same techniques apply to all programmable calculators, although program **syntax** is almost certainly different for different models of calculator, even from the same manufacturer.

(40) Every programmable calculator has some sort of **programming language** stored in memory.

(41) Programmable calculators from Texas Instruments usually have two internal programming languages: **Assembly Language**, and **BASIC**.

Getting Help Running BASIC on the TI83

(42) I got lots of help learning how to use BASIC from a TI calculator user's group located on the internet as **ticalc.org**.

(43) This website has information and tutorials for learning how to run both assembly language and BASIC on different types of TI calculators.

(44) If you have a serial cable for your calculator, you can also download an **emulator** for your calculator which allows you to run a **virtual calculator** on your computer.

(45) An emulator **makes learning programming easier** by allowing you to type in programs using your computer, instead of the little buttons on your calculator.

(46) There are other useful websites including the official Texas Instruments website.

Writing Your First Program

(47) To start writing a program, press the PRGM button on your calculator. You should see the words EXEC EDIT NEW.

(48) Use the cursor buttons to highlight NEW and then press ENTER.

- (49) Name your program using the alphabet keys. Program names must be eight or less characters.
- (50) Once you have entered a name, the cursor will appear next to a colon where you will write in the first line of code.
- (51) The easiest program to write is one that just displays a message on the screen. Press the PRGM button again and you will see the words CTL I/O EXEC.
- (52) Highlight the I/O, scroll down to highlight **Disp** and press ENTER.
- (53) Now, back in the program, type in the word “Hello” with double quotes
- (54) To exit the program editor, press 2ND QUIT.
- (55) Back out in the main screen, press PRGM, select your program and press ENTER.

Assigning Variables

- (56) In most mathematic forms you store a value in a variable by setting the variable equal to the value such as $A = 1$.
- (57) In TI BASIC, you use the STO→ button to assign a value to a variable. The following syntax $1 \rightarrow A$ assigns the value 1 to the variable A.
- (58) Likewise to say the F is equal to $A + BX + CX^2 + DX^3$ you would write the phrase,...

$$A + BX + CX^2 + DX^3 \rightarrow F$$

(The STO→ button is located on the calculator itself gives you the → symbol)

- (59) You can also prompt the user for enter a value in a variable by using the INPUT and PROMPT commands found in the I/O menu.

INPUT “ENTER A: “, A or PROMPT A

SOLVING FOR THE ROOT OF A CUBIC EQUATION

- (60) Try typing in program on the next page to solve for the roots of a cubic equation using the Newton-Raphson method. The code demonstrates several additional features such as DO WHILE loops. Start by naming your program ROOTS.

```

: ClrHome
: Disp "THIS PROGRAM"
: Disp "SOLVES FOR THE"
: Disp "ROOTS OF A CUBIC"
: Disp "EQUATION"
: Disp " "
: Disp " AX3 + BX2 + CX + D"
: Disp " "
: Disp "PRESS ENTER"
: Pause
: Disp "ENTER A B C D AND X"
: Prompt A
: Prompt B
: Prompt C
: Prompt D
: Prompt X
: A*X3 + B*X2 + C*X + D STO→ F
: 3*A*X2 + 2*B*X + C STO→ R
: X - (F/R) STO→ N
: While abs(N-X) > 0.00001
: N → X
: A*X3 + B*X2 + C*X + D STO→ F
: 3*A*X2 + 2*B*X + C STO→ R
: X - (F/R) STO→ N
: End
: Disp "THE ROOT"
: Disp "CLOSEST"
: Disp "TO X IS: "
: Disp N

```

(61) The While, If, Then, and End commands are found by pressing PRGM.

(62) The ³ is found by pressing MATH, and the > symbol by pressing 2nd-MATH.

(63) You run the program in the main screen. Enter the coefficients 2 4 6 8 and the guess 1. Your answer should be -.650295862. How would we generate a complex root?

Friday and Monday

(64) Friday is the last day of lab. We will use the lab time in Friday for anyone who wants to bring in their Matlab, VBA, and MathCad programs to make sure they work for the final exam. Monday is the last day of class for us this semester. We will use the class time on Monday for an in-depth review of the material that will be covered on the final exam.