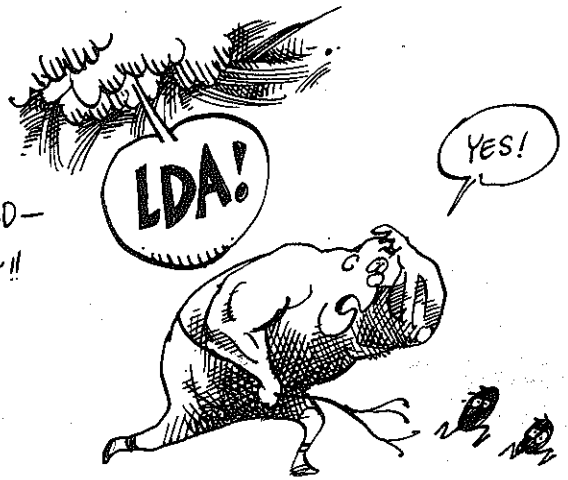
SO YOU SEE, CONTROL IS NO TYRANT AT ALL. IT ONLY DOES WHAT IT'S TOLD— COMPLETELY AUTOMATICALLY!!

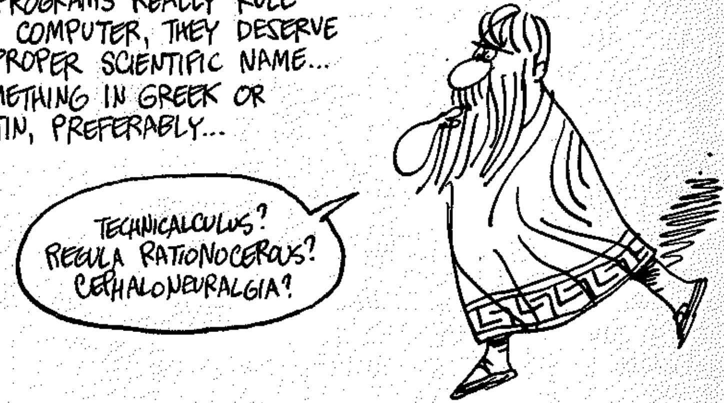


IF YOU REALLY WANT TO IMAGINE THE CONTROL SECTION'S PERSONALITY, THINK OF A PERFECTLY EFFICIENT *BUREAUCRAT*, ACTING IN STRICT OBEDIENCE TO THE COMPUTER'S *REAL* BOSS: THE PROGRAM!

PART III

SOFTWARE

IF PROGRAMS REALLY RULE THE COMPUTER, THEY DESERVE A PROPER SCIENTIFIC NAME... SOMETHING IN GREEK OR LATIN, PREFERABLY...



* *

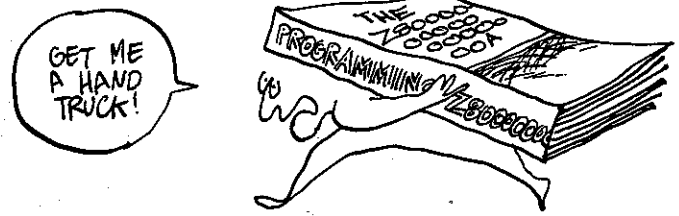
BUT THAT'S NOT HOW IT IS IN COMPUTER SCIENCE... INSTEAD, PROGRAMS IN GENERAL ARE CALLED **SOFTWARE**,

TO DISTINGUISH THEM FROM THE CIRCUIT BOARDS, CATHODE RAY MONITORS, DISK DRIVES, KEYBOARDS, AND OTHER ITEMS OF COMPUTER **HARDWARE**.

HARDWARE SOFTWARE TUPPERWARE

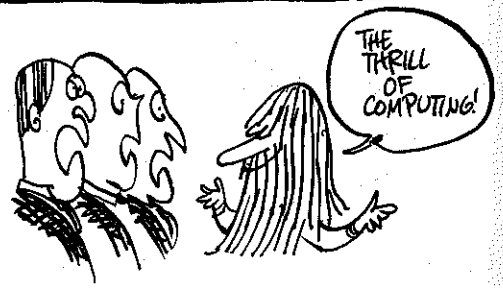
WHAT'S REALLY FUNNY ABOUT THE NAME IS THAT SOFTWARE IS ONE OF THE ***HARDEST*** THINGS ABOUT COMPUTING!

WHILE HARDWARE HAS BEEN DROPPING IN PRICE AND GROWING IN POWER, SOFTWARE ONLY GETS MORE HORRENDOUSLY COMPLEX!



⇒ WE SEE SMALLER AND SMALLER CHIPS WITH BIGGER + BIGGER MANUALS!

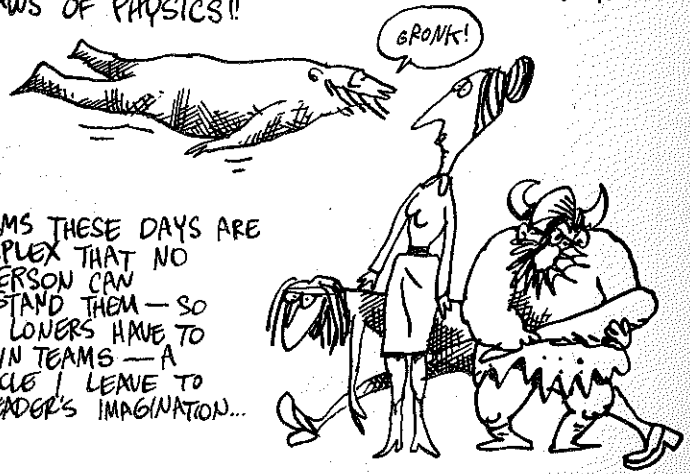
IT'S OFTEN IMPOSSIBLE TO ESTIMATE HOW MUCH TIME, MONEY, AND AGONY A GIVEN SOFTWARE PROBLEM WILL COST TO SOLVE... WHAT A WAY TO RUN A BUSINESS!



LIKEWISE THERE'S A DIFFERENCE BETWEEN THE IMAGE OF HARDWARE AND SOFTWARE WORKERS —

HARDWARE TYPES ARE ENGINEERS... INTO GADGETS... MOSTLY MEN... BOUND BY THE LAWS OF PHYSICS...

PROGRAMMERS HAVE NO TOOL BUT THEIR BRAINS... THEY'RE MORE OFTEN WOMEN... SUPPOSED TO BE SOLITARY DREAMERS WHOSE IDEAS HAVE NOTHING TO DO WITH THE LAWS OF PHYSICS!!



PROGRAMS THESE DAYS ARE SO COMPLEX THAT NO ONE PERSON CAN UNDERSTAND THEM — SO THESE LONERS HAVE TO WORK IN TEAMS — A SPECTACLE I LEAVE TO THE READER'S IMAGINATION...

WHILE ADA LOVELACE WAS THE ORIGINAL PROGRAMMER, THE FIRST PERSON TO PROVE THE FULL POWER OF SOFTWARE WAS

ALAN TURING

(1912-1954)

TURING, WHO ENJOYED LONG-DISTANCE RUNNING BACK WHEN THAT WAS CONSIDERED WEIRD, PROBABLY WENT INTO COMPUTERS TO SHRINK THE SIZE OF HIS JOGGING CLOCK.

IN 1936 HE DREAMED UP THE *TURING MACHINE*...

TURING MACHINES AREN'T REAL MACHINES... THEY'RE ABSTRACT MACHINES, EXISTING ONLY IN THEORY...

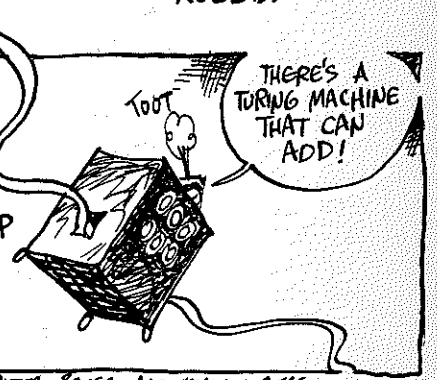
ROUGHLY SPEAKING, A TURING MACHINE IS AN INPUT-OUTPUT DEVICE: A BLACK BOX THAT READS A SEQUENCE OF 0's AND 1's.

THE OUTPUT DEPENDS ONLY ON THE PRESENT INPUT (0 OR 1) AND THE PREVIOUS OUTPUT.

THE NATURE OF THE OUTPUT IS UNIMPORTANT.

THE MAIN THING IS THAT THE CHANGES FROM ONE OUTPUT STATE TO THE NEXT ARE GIVEN BY DEFINITE RULES, CALLED THE *TRANSITION RULES.*

THE REASON TURING MACHINES ARE IMPORTANT IS THAT THEY ARE A A WAY OF THINKING PHYSICALLY ABOUT LOGIC. ANY WELL-DEFINED, STEP-BY-STEP LOGICAL PROCEDURE CAN BE EMBODIED IN SOME TURING MACHINE.

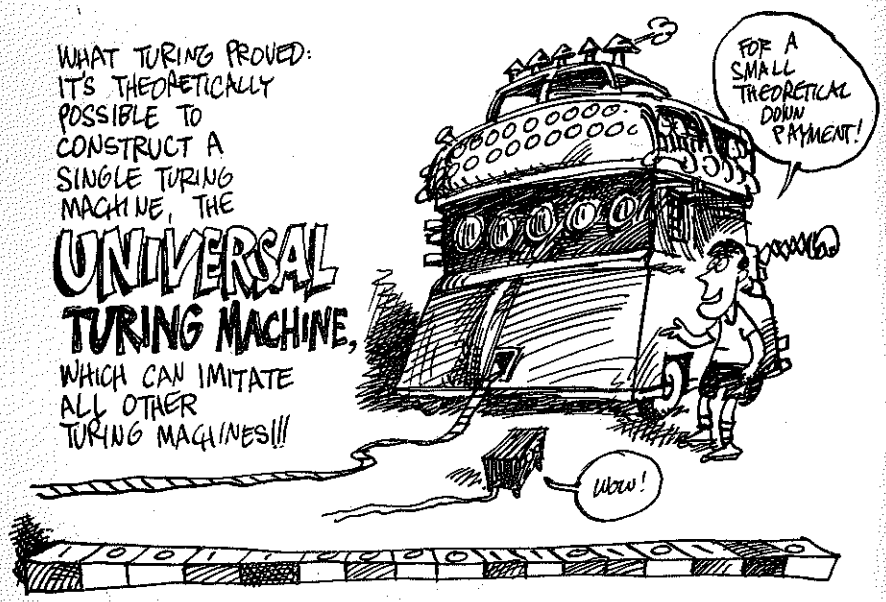


*FOR DETAILS, SEE J. WEIZENBAUM'S COMPUTER POWER AND HUMAN REASON, CHAPTER 2.

WHAT TURING PROVED: IT'S THEORETICALLY POSSIBLE TO CONSTRUCT A SINGLE TURING MACHINE, THE

UNIVERSAL TURING MACHINE,

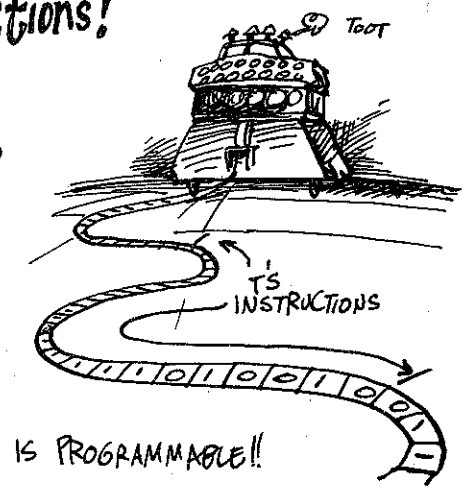
WHICH CAN IMITATE ALL OTHER TURING MACHINES!!!



THE TRICK IS THAT THE UNIVERSAL TURING MACHINE CAN...

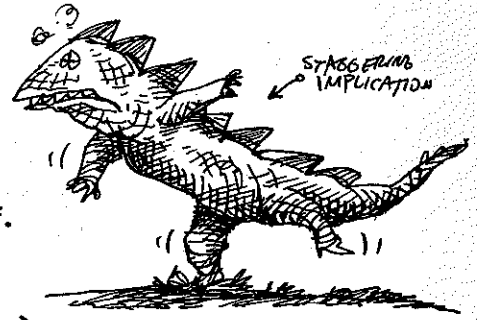
⇒ read instructions!

THAT IS, TO MAKE THE UNIVERSAL TURING MACHINE (**U**) ACT LIKE MACHINE **T**, YOU ENCODE **T**'S TRANSITION RULES ONTO **U**'S TAPE. AT EACH STEP, **U** OBSERVES ITS OWN INPUT, THEN REFERS TO **T**'S TRANSITION RULES TO SEE WHAT TO DO.



⇒ IN OTHER WORDS, **U** IS PROGRAMMABLE!!

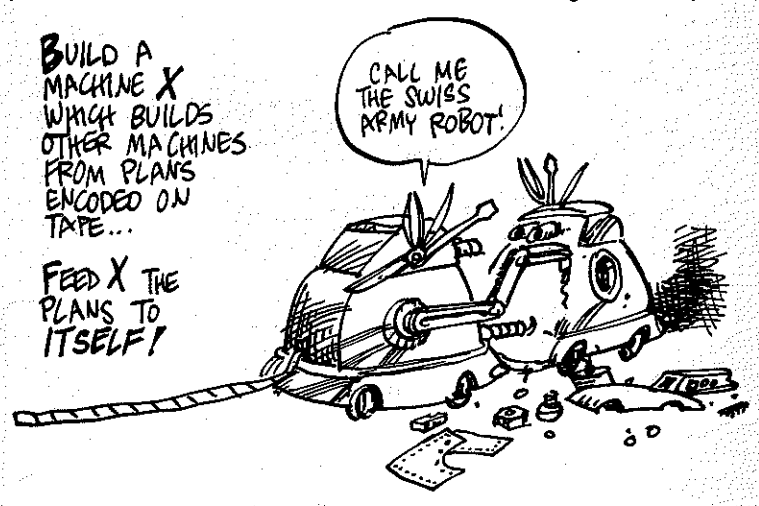
THE IMPLICATIONS ARE STAGGERING: A SINGLE, PROGRAMMABLE MACHINE CAN PERFORM **ANY** WELL-DEFINED, STEP-BY-STEP LOGICAL PROCEDURE. (REMEMBER, TURING SAW THIS TEN YEARS BEFORE A REAL COMPUTER WAS BUILT.)



JOHN VON NEUMANN CARRIED TURING'S IDEAS A STEP FURTHER. VON NEUMANN REALIZED THAT ONE COULD:

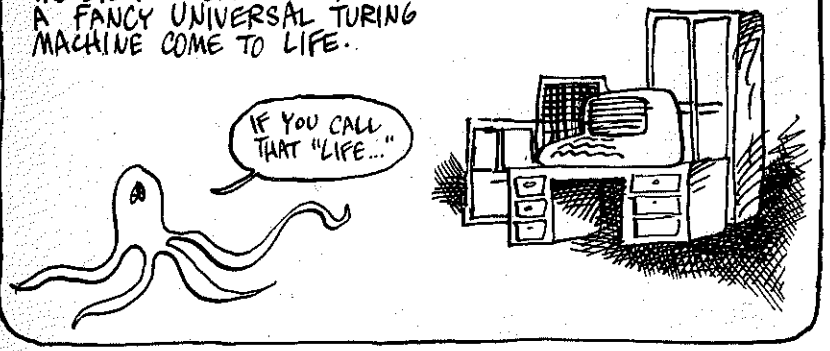
BUILD A MACHINE **X** WHICH BUILDS OTHER MACHINES FROM PLANS ENCODED ON TAPE...

FEED **X** THE PLANS TO **ITSELF!**

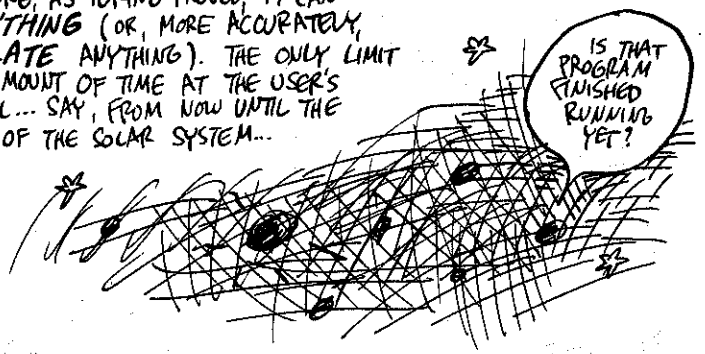


⇒ **SELF-REPRODUCING MACHINES** ARE POSSIBLE!!

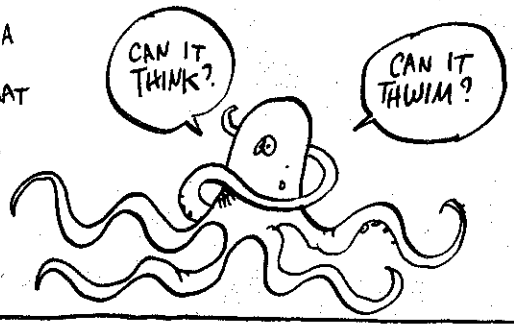
THE DIGITAL COMPUTER IS A FANCY UNIVERSAL TURING MACHINE COME TO LIFE.



THEREFORE, AS TURING PROVED, IT CAN DO **ANYTHING** (OR, MORE ACCURATELY, **SIMULATE** ANYTHING). THE ONLY LIMIT IS THE AMOUNT OF TIME AT THE USER'S DISPOSAL... SAY, FROM NOW UNTIL THE DEATH OF THE SOLAR SYSTEM...



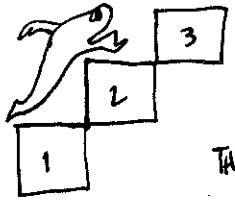
TO BE PERFECTLY HONEST, THERE ARE A COUPLE OF OTHER QUALIFICATIONS ON THAT "ANYTHING." WHAT KIND OF "ANYTHING" CAN A COMPUTER DO?



IN A WORD, COMPUTERS DO

ALGORITHMS

AN ALGORITHM IS SIMPLY ANY WELL DEFINED, STEP-BY-STEP PROCEDURE: A RECIPE, IF YOU WILL!



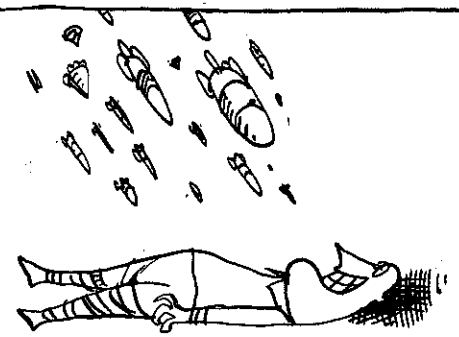
STEP-BY-STEP, MEANING EACH STEP IS COMPLETED BEFORE THE NEXT IS BEGUN.

WELL DEFINED, MEANING EACH STEP IS COMPLETELY DETERMINED BY CURRENT INPUT AND THE RESULTS OF PREVIOUS STEPS. NO AMBIGUITY ALLOWED!

EXAMPLES OF ALGORITHMS:

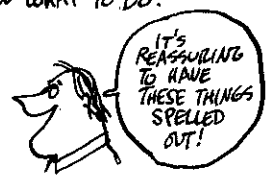
"IF NUCLEAR WARHEADS ARE FALLING LIKE HAILSTONES, I WILL LIE DOWN AND TRY TO ENJOY IT.

OTHERWISE, I WILL GO TO WORK AS USUAL."



IT'S AN ALGORITHM BECAUSE I ALWAYS KNOW WHAT TO DO:

1. CHECK TO SEE IF WARHEADS ARE FALLING
2. IF YES, LIE DOWN + ENJOY!
3. IF NO, GO TO WORK.



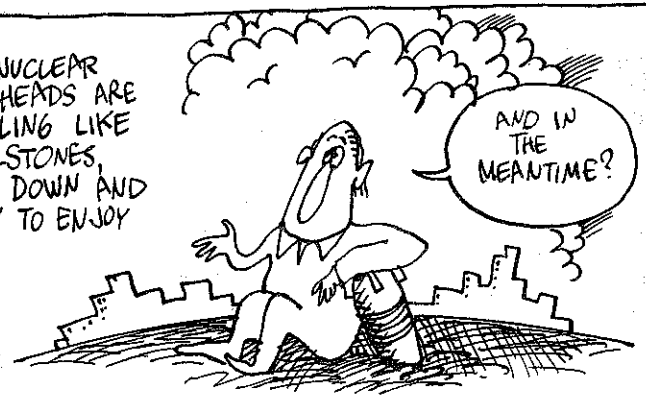
LIKEWISE, ALGEBRAIC FORMULAS REPRESENT ALGORITHMS

$y = x^2 + 2x + 10$ MEANS —

(1) INPUT A NUMBER x
(2) MULTIPLY x TIMES ITSELF
(3) MULTIPLY x TIMES 2
(4) ADD THE RESULTS OF (2) AND (3)
(5) ADD 10 TO THE RESULT OF (4)

EXAMPLES OF NON-ALGORITHMS:

"IF NUCLEAR WARHEADS ARE FALLING LIKE HAILSTONES, LIE DOWN AND TRY TO ENJOY IT."



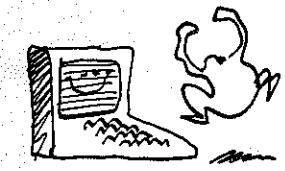
THIS FAILS TO TELL YOU WHAT TO DO IF NO WARHEADS ARE FALLING... SO IT'S NOT WELL DEFINED.

ANOTHER?

HOW ABOUT

$y = x^2 ++ 2x - 10$?

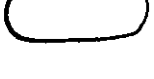
THIS IS NO ALGORITHM BECAUSE IT'S NOT EXPRESSED IN PROPER "ALGEBRAIC GRAMMAR." WE ASSIGN NO MEANING TO THE SYMBOLS "++".



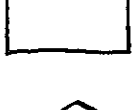
IF YOU TRY TO MAKE A COMPUTER DO A NON-ALGORITHM, IT WILL JUST SIT THERE FLASHING ERROR MESSAGES!

SOME STANDARD SYMBOLS ARE USED TO MAKE ALGORITHMS EASIER TO FOLLOW. EACH STEP IS REPRESENTED BY A SPECIALLY SHAPED — WHAT ELSE? — BOX. THE SHAPE INDICATES WHAT TYPE OF STEP IS TO BE EXECUTED:

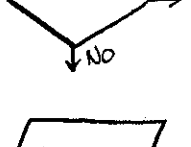
THERE ARE 4 POSSIBILITIES:



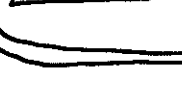
BEGIN OR END



PERFORM A PROCEDURE (ADD, SUBTRACT, ETC.)



CONDITIONAL BRANCH

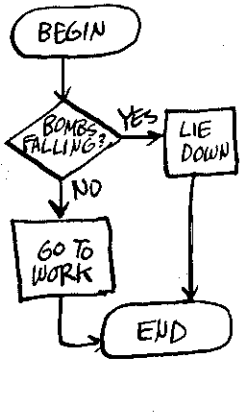


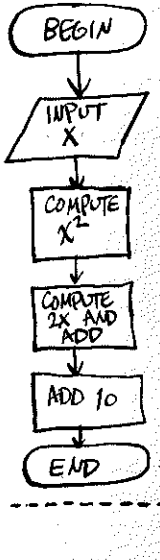
INPUT OR OUTPUT

THE "FLOW" OF THE ALGORITHM IS REPRESENTED BY ARROWS →, AND WHEN ALL THE SYMBOLS ARE COMBINED, IT'S A

FLOW CHART

HERE ARE THE FLOW CHARTS OF THE ALGORITHMS FROM A COUPLE OF PAGES BACK:

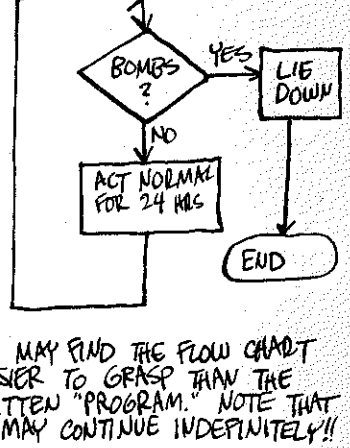




IN BOTH ALGORITHMS, THE FLOW PROCEEDS IN ONE DIRECTION, FROM START TO FINISH.

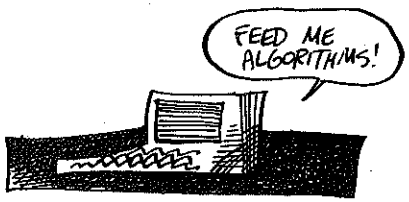
IT'S ALSO POSSIBLE FOR THE FLOW OF ALGORITHMS TO JUMP FORWARD OR BACKWARD. FOR EXAMPLE, LET'S REWRITE THAT FIRST ALGORITHM:

1. IF BOMBS ARE FALLING, GO TO STEP 2. OTHERWISE, GO TO STEP 4.
2. LIE DOWN AND ENJOY!
3. GO TO STEP 6.
4. LEAD A NORMAL LIFE FOR 24 HOURS
5. GO TO STEP 1
6. END



YOU MAY FIND THE FLOW CHART EASIER TO GRASP THAN THE WRITTEN "PROGRAM." NOTE THAT IT MAY CONTINUE INDEFINITELY!!

FLOW CHARTS ARE USEFUL IN HELPING TO DESIGN ALGORITHMS — SIMPLE ONES, ANYWAY — AND DESIGNING ALGORITHMS IS WHAT COMPUTER PROGRAMMING IS ALL ABOUT!!



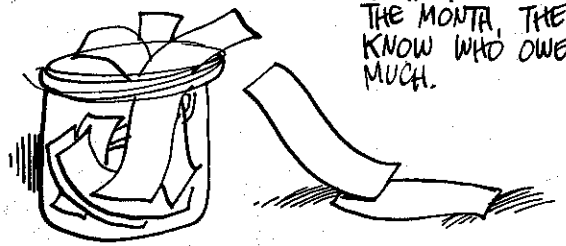
THE FIRST STEP IN WRITING ANY PROGRAM IS TO **ANALYZE** THE JOB TO BE DONE, AND SEE HOW TO DO IT ALGORITHMICALLY!

FAILURE TO THINK ALGORITHMICALLY HAS CAUSED MANY SOFTWARE NIGHTMARES!! MOST SOFTWARE DESIGNERS HAVE HORROR STORIES ABOUT CUSTOMERS WHO DIDN'T KNOW **EXACTLY** WHAT THEY WANTED!!

LET'S TRY A COUPLE MORE EXAMPLES... A LITTLE MORE LIKE WHAT A COMPUTER MIGHT ACTUALLY BE ASKED TO DO...

"ROOMMATE RECEIPTS"

TWO ROOMATES, LISA AND SOPHIE, SHARE THEIR MEALS. THEY BOTH SHOP FOR FOOD AND SAVE THEIR RECEIPTS. AT THE END OF THE MONTH, THEY WANT TO KNOW WHO OWES WHOM HOW MUCH.



"MULTIPLE PLUG-INS"

THIS ONE ASKS THE COMPUTER TO EVALUATE THE EXPRESSION

$$X^2 + 2X + 10$$

NOT JUST AT ONE VALUE OF X, BUT FOR MANY VALUES, NAMELY

X = 0, 0.1, 0.2, 0.3, ... AND SO ON... UP TO 2.0.

FOR "ROOMMATE RECEIPTS" WE REASON LIKE SO:

LET S = SOPHIE'S EXPENSES
L = LISA'S EXPENSES

THEN THE TOTAL EXPENSE IS S+L, AND EACH ROOMMATE'S SHARE IS

$\frac{1}{2}(S+L)$.

IF LISA OUTSPENT SOPHIE, SO $L > S$*, THEN SOPHIE OWES LISA

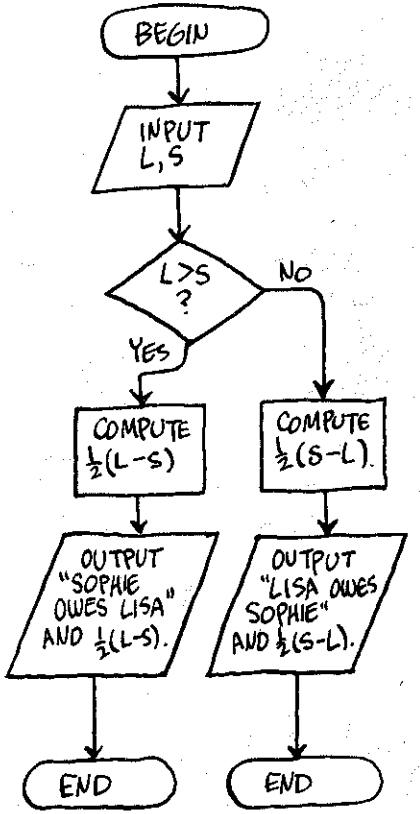
$\frac{1}{2}(S+L) - S$, OR

$\frac{1}{2}(L-S)$.

OTHERWISE (WHEN $S \geq L$*), LISA OWES SOPHIE

$\frac{1}{2}(S-L)$.

THE ALGORITHM'S OUTPUT IS TO TELL US WHO OWES WHOM AND HOW MUCH.



* > MEANS "IS GREATER THAN"; ≥ MEANS "IS GREATER THAN OR EQUAL TO"; < MEANS "IS LESS THAN"; ≤ MEANS "IS LESS THAN OR EQUAL TO".

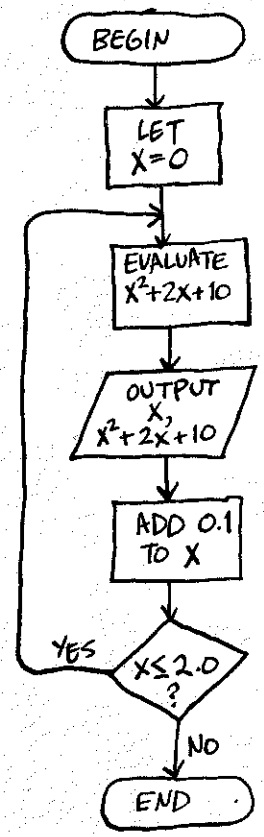
IN "MULTIPLE PLUG-INS," WE WANT TO EVALUATE A SINGLE EXPRESSION, $x^2+2x+10$, REPEATEDLY AT DIFFERENT VALUES OF X (NAMELY 0.0, 0.1, 0.2, ... , 1.9, 2.0)

THE CORE OF THE ALGORITHM WILL BE THIS LOOP:

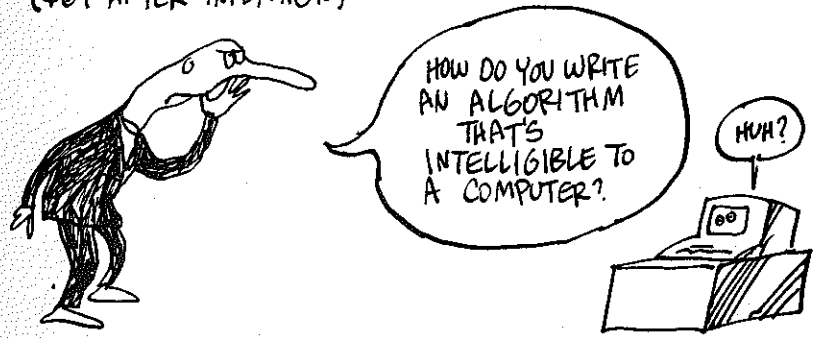
1. PLUG THE CURRENT VALUE OF X INTO $x^2+2x+10$
2. PRINT THE RESULT
3. NEXT X
4. RETURN TO STEP 1.

WE ALSO HAVE TO SPECIFY WHAT X TO START WITH, WHEN TO STOP, AND HOW TO COMPUTE "NEXT X."

NOTE HOW THE FLOW CHART SHOWS HOW THE PROGRAM LOOPS BACK, PLUGGING IN SUCCESSIVE VALUES OF X UNTIL X EXCEEDS 2.

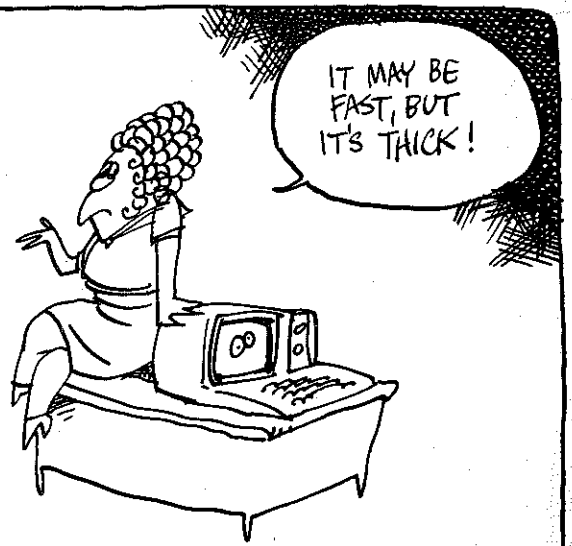


NOW THE $738 QUESTION: ($64 AFTER INFLATION):



IN OTHER WORDS, HOW DO YOU PROGRAM A COMPUTER?

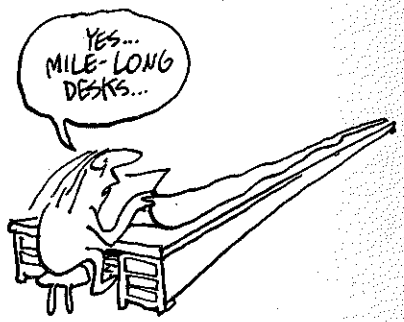
UNFORTUNATELY, YOU HAVE TO SPEAK THE COMPUTER'S LANGUAGE— BECAUSE THE COMPUTER IS STILL TOO STUPID TO UNDERSTAND YOURS!



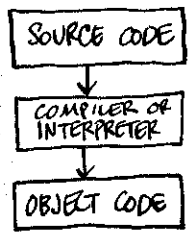
WHAT LANGUAGE DOES THE COMPUTER UNDERSTAND?

AT THE VERY BEGINNING, PROGRAMMERS WROTE DIRECTLY IN "MACHINE LANGUAGE"—BINARY CODE. THIS WAS OBVIOUSLY A HEADACHE!

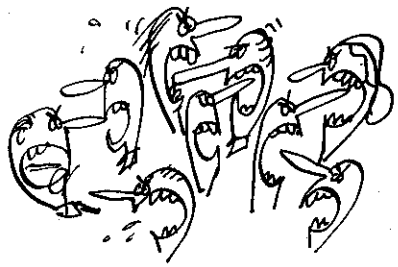
SOON THEY SWITCHED TO ASSEMBLY LANGUAGE (SEE P. 174), AIDED BY AUTOMATIC "ASSEMBLERS," WHICH TRANSLATED ASSEMBLY LANGUAGE MNEMONICS INTO MACHINE CODE. STILL SOMETHING MORE WAS NEEDED!



AND FINALLY, THE **HIGHER-LEVEL** PROGRAMMING LANGUAGES WERE INVENTED. THESE CONTAIN FAMILIAR ENGLISH-LIKE COMMANDS, SUCH AS "PRINT," "READ," AND "DO," WHICH ARE TRANSLATED INTO MACHINE LANGUAGE BY COMPLEX PROGRAMS CALLED COMPILERS OR INTERPRETERS. HIGHER-LEVEL PROGRAMS ARE SOMETIMES CALLED "SOURCE CODE," AND THE MACHINE-LANGUAGE TRANSLATION IS CALLED "OBJECT CODE."



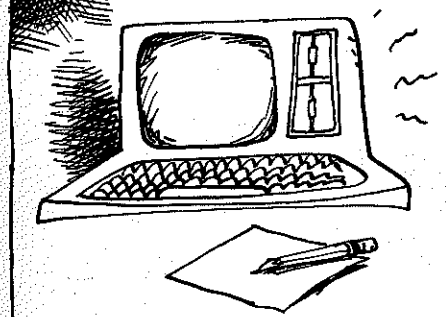
THE FIRST HIGHER-LEVEL LANGUAGE WAS **FORTRAN** ("FORMULA TRANSLATOR"), WHICH MADE ITS DEBUT IN THE EARLY 1950's. SINCE THEN, LITERALLY HUNDREDS OF LANGUAGES HAVE BEEN WRITTEN, EACH WITH ITS OWN ARMY OF RABID DEVOTEES!



WE'RE GOING TO TAKE A QUICK LOOK AT **BASIC** — **B**EGINNER'S **A**LL-PURPOSE **S**YMBOLIC **I**NSTRUCTION **C**ODE. BASIC IS EASY TO LEARN AND WIDELY USED, DESPITE CRITICISM (ESPECIALLY BY PASCAL ADMIRERS) THAT IT PROMOTES "BAD PROGRAMMING HABITS."

WITH APOLOGIES TO PASCAL, THEN, HERE'S A LITTLE BASIC...

BASIC BASIC



THERE ARE TWO WAYS TO WRITE A BASIC PROGRAM: WITH PENCIL AND PAPER, OR DIRECTLY AT THE COMPUTER.

IT'S GOOD PRACTICE TO PLAN PROGRAMS ON PAPER FIRST, TO WORK OUT THE ESSENTIAL IDEAS AND STRUCTURE. BUT EVENTUALLY YOU MUST SIT DOWN AT THAT KEYBOARD!

SOME MACHINES ARE READY FOR BASIC AS SOON AS YOU TURN THEM ON. OTHERS ONLY BRING IT UP ON COMMAND. IF IN DOUBT, ASK!

WHEN THE COMPUTER IS READY, IT GIVES YOU A "PROMPT" OF SOME KIND: THE WORD "READY" OR JUST THE SIGN ">".

THE COMPUTER KEYBOARD RESEMBLES A STANDARD TYPEWRITER'S "QWERTY" KEYBOARD... EXCEPT THAT AS YOU TYPE, CHARACTERS APPEAR ON THE CRT (CATHODE RAY TUBE) SCREEN, INSTEAD OF ON PAPER. TO GO TO THE NEXT LINE, HIT THE **RETURN** (↵) KEY. HERE'S A SIMPLE BASIC PROGRAM:

```
10 REM BASIC MULTIPLICATION
20 READ A, B
30 DATA 5.6, 1.1
40 LET C=A*B
50 PRINT "THE PRODUCT IS"; C
60 END
```

BASIC MATH:
A+B, A-B } AS USUAL
A*B... A TIMES B
A/B... A DIVIDED BY B
A↑B... A TO THE Bᵀᴴ POWER

THE PROGRAM IS NOW STORED IN MEMORY. TO RUN IT, TYPE "RUN", FOLLOWED BY THE RETURN KEY. THE SCREEN DISPLAYS:

```
RUN
THE PRODUCT IS 6.16
```

A FEW POINTS TO NOTE:

> EVERY LINE BEGINS WITH A **LINE NUMBER** (10, 20, ...). EVERY LINE OF A BASIC PROGRAM MUST HAVE A NUMBER! IT'S WISE TO COUNT BY TENS, SO YOU CAN INSERT LINES LATER.

> THE FIRST LINE (10) IS A **REMARK.** REMARKS EXPLAIN THE PROGRAM BUT AREN'T EXECUTED BY THE COMPUTER. THE PREFIX "REM" IDENTIFIES REMARKS. WE MIGHT INSERT ONE HERE:

```
20 READ A, B
25 REM THESE ARE THE #S TO BE MULT'D
30 DATA 5.6, 1.1
```

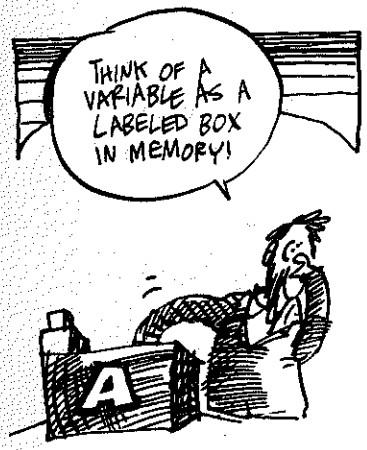
> PROGRAM **STATEMENTS** CONSIST OF **INSTRUCTIONS** ("LET", ETC), **NUMBERS** (5.6, 1.1), **VARIABLES** (A, B, C), **TEXT** ("THE PRODUCT IS"), AND **PUNCTUATION.**

```
50 PRINT "THE PRODUCT IS"; C
```

QUOTES SPACES SEMICOLON

> EACH OF THESE HAS A PRECISE MEANING!

"NUMERICAL" VARIABLES



A NUMERICAL VARIABLE IN BASIC IS LIKE A VARIABLE IN ALGEBRA. IT ASSUMES A NUMERICAL VALUE, WHICH MAY VARY (BUT IT HAS ONLY ONE VALUE AT A TIME!). ONLY THESE SYMBOLS CAN BE USED AS VARIABLES:

A, B, C, D, ...		...Z
A0, B0, ...	AND	...Z0
A1, B1, ...	EVERYTHING IN BETWEEN!	...Z1
⋮		⋮
A9, B9, ...		...Z9

THERE ARE SEVERAL WAYS TO ASSIGN A VALUE TO A VARIABLE: ONE IS THE **READ-DATA** STATEMENT:

```
20 READ A,B
30 DATA 5.6, 1.1
```

COMMAS ARE ESSENTIAL!!

THIS INSTRUCTS THE COMPUTER TO ASSIGN THE NUMERICAL VALUES IN THE DATA STATEMENT — IN ORDER — TO THE VARIABLES IN THE READ STATEMENT.

```
20 READ A, B, C
30 DATA 5.6, 1.1
```

THIS IS A **BUG!**

DISGUSTING!

ANOTHER WAY TO ASSIGN VALUES TO VARIABLES IS WITH

LET

```
10 LET Q=6.5
20 LET R=2*Q
30 LET S=Q↑2+R+10
```

MAKES R=13

MAKES S= $(6.5)^2+13+10$ = 65.25

THE LET STATEMENT ASSIGNS THE VALUE ON THE **RIGHT** OF THE EQUALITY SIGN, "=", TO THE VARIABLE ON THE **LEFT.** THE RIGHT-HAND SIDE MAY BE A NUMBER, OR SOME MATHEMATICAL EXPRESSION INVOLVING OTHER VARIABLES — AS LONG AS THEY ALREADY HAVE VALUES!!

```
10 LET Q=6.5
20 LET Q=0.5*R
30 LET S=Q↑2+R+10
```

HERE STATEMENT 20 DOES NOT ASSIGN ANY VALUE TO R, BECAUSE R IS NOT ON THE LEFT SIDE OF "=". IN FACT, IF R HASN'T BEEN ASSIGNED SOME VALUE EARLIER IN THE PROGRAM, THEN STATEMENT 20 GIVES Q AN INDETERMINATE VALUE! BUT—

```
10 LET M=0
20 LET M=M+1
30 LET M=M+1
```

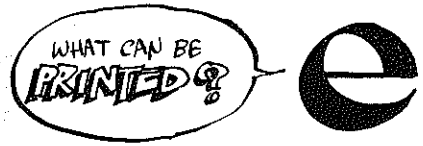
MAKES M=1

MAKES M=2

THESE STRANGE-LOOKING STATEMENTS ARE PERFECTLY O.K! "LET M=M+1" MEANS "ASSIGN TO THE VARIABLE M A VALUE EQUAL TO ITS CURRENT VALUE PLUS 1."

PRINT

THIS IS AN OUTPUT COMMAND, MEANING "DISPLAY ON THE SCREEN," NOT "PRINT ON PAPER."



YOU CAN PRINT ANY TEXT:

```
10 PRINT "ANY NUKES TODAY?"
RUN
ANY NUKES TODAY?
```

QUOTATION MARKS ESSENTIAL!

QUOTATION MARKS REMOVED

PRINT A VARIABLE AND YOU GET ITS VALUE:

```
10 LET X=77001
20 PRINT X
RUN
77001
```

BUT—

```
10 LET X=77001
20 PRINT "X"
RUN
X
```

QUOTATION MARKS MAKE THE COMPUTER TREAT X AS A TEXT.

PRINT A MATHEMATICAL EXPRESSION AND YOU GET ITS VALUE:

```
10 LET Z=1.5
20 PRINT Z↑2+2*Z+10
RUN
15.25
```

BECAUSE $(1.5)^2 + 2 \times 1.5 + 10 = 2.25 + 3.0 + 10 = 15.25$

SEMICOLON (;)

A SEMICOLON AFTER A PRINT STATEMENT CAUSES THE *NEXT* PRINT STATEMENT TO DISPLAY ITS OUTPUT ON THE SAME LINE AND DIRECTLY AFTER THE FIRST ONE'S:

```
10 LET A=1
20 PRINT "INFINITY IS MORE THAN";
30 PRINT A
RUN
INFINITY IS MORE THAN 1
```

IT'S O.K. TO ABBREVIATE THIS:

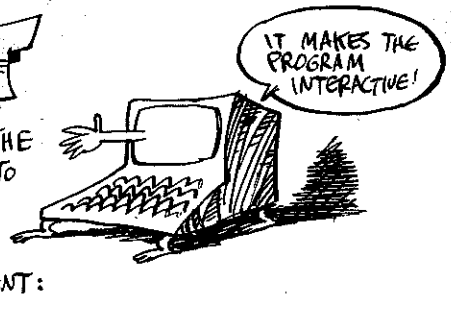
```
10 LET A=1
20 PRINT "INFINITY IS MORE THAN"; A
RUN
INFINITY IS MORE THAN 1
```

FOR EXAMPLE, WE COULD REWRITE THE PROGRAM ON P. 208.

```
10 REM BASIC MULTIPLICATION
20 READ A,B
30 DATA 5.6, 1.1
40 LET C=A*B
50 PRINT "THE PRODUCT OF "; A; "AND"; B; "IS"; C; "."
60 END
RUN
THE PRODUCT OF 5.6 AND 1.1 IS 6.16.
```

THERE ARE ALSO SOME NIFTY TRICKS USING THE COMMA AND PRINT, BUT WE WON'T GET INTO IT...

INPUT



THIS STATEMENT ALLOWS THE USER TO ASSIGN VALUES TO VARIABLES WHILE THE PROGRAM IS RUNNING.

THE FORM OF THE STATEMENT:

```
INPUT A
```

WHEN THE PROGRAM RUNS AND REACHES AN INPUT STATEMENT, THE SCREEN DISPLAYS:

```
?
```

THIS INDICATES THAT THE PROGRAM HAS HALTED, AWAITING INPUT. YOU TYPE SOME NUMBER (FOLLOWED BY "RETURN," AS ALWAYS!):

```
5.6
```

AND THE PROGRAM CONTINUES RUNNING.
"INPUT" AND "PRINT" CAN BE USED IN COMBINATION TO LET YOU KNOW WHAT SORT OF INPUT IS EXPECTED:

```
10 BASIC DIVISION
20 PRINT "TYPE THE NUMERATOR."
30 INPUT N
40 PRINT "TYPE THE NON-ZERO DENOMINATOR."
50 INPUT D
60 PRINT N;"/";D;"=";N/D
70 END
RUN
TYPE THE NUMERATOR.
? 5
TYPE THE NON-ZERO DENOMINATOR.
? 8
5/8 = 0.625
```

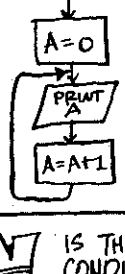
TYPED BY THE USER.

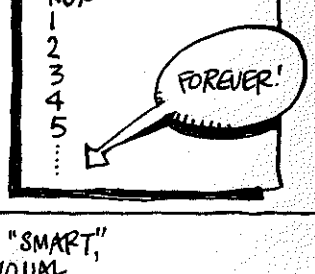
GOTO

THIS IS THE UNCONDITIONAL BRANCHING INSTRUCTION.

"GO TO (LINE NUMBER)" TRANSFERS CONTROL TO A LINE OTHER THAN THE NEXT. THE PROGRAM THEN CONTINUES FROM THERE, AS IN THIS ENDLESS LOOP:

```
10 LET A=0
20 PRINT A
30 LET A=A+1
40 GO TO 20
```





IF-THEN

IS THE "SMART," CONDITIONAL JUMP.

IT HAS THE GENERAL FORM
IF (CONDITION) **THEN** (LINE NUMBER).
THE CONDITION HAS THE FORM:

NUMERICAL EXPRESSION { = , < , <= , > , >= , <> }* NUMERICAL EXPRESSION

AS IN

```
IF A<=B THEN 30
```

THIS ALWAYS INCLUDES THE UNSTATED INSTRUCTION, "OTHERWISE, GO TO THE NEXT LINE."

```
10 LET A=0
20 PRINT A
30 LET A=A+1
40 IF A<=2 THEN 20
50 END
RUN
0
1
2
```

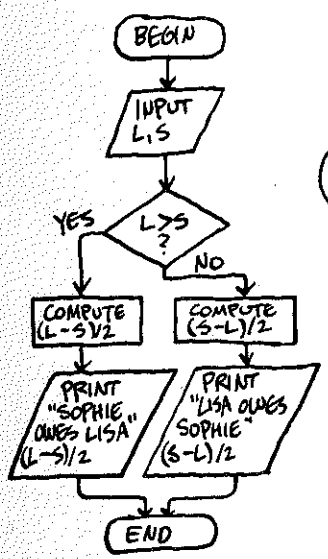
"OTHERWISE, NEXT LINE!"

* < LESS THAN, <= LESS THAN OR EQUAL TO, > GREATER THAN, >= GREATER THAN OR EQUAL TO, <> DOES NOT EQUAL.

THIS IS ENOUGH TO WRITE BASIC PROGRAMS FOR THE TWO ALGORITHMS FROM P. 201:

ROOMMATE RECEIPTS

THE FLOW CHART:



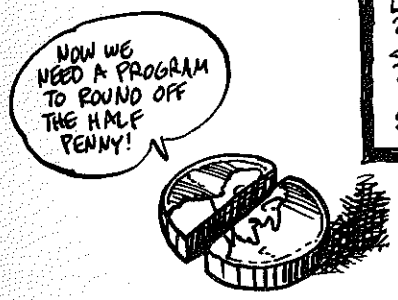
THE PROGRAM:

```
10 PRINT "LISA SPENT"
20 INPUT L
30 PRINT "SOPHIE SPENT"
40 INPUT S
50 IF L>S THEN 80
60 PRINT "LISA OWES SOPHIE"; (S-L)/2
70 GO TO 90
80 PRINT "SOPHIE OWES LISA"; (L-S)/2
90 END
```

SEE HOW "IF-THEN" AND "GO TO" ARE USED? IF L>S, THEN LINES 60 AND 70 ARE NOT EXECUTED. OTHERWISE, THEY ARE EXECUTED, AND LINE 70 ENSURES THAT LINE 80 IS SKIPPED.

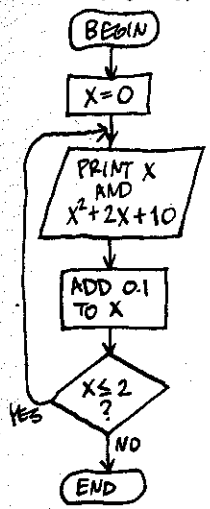
IF THE PROGRAM IS RUN:

```
RUN
LISA SPENT
? 93.75
SOPHIE SPENT
? 77.38
SOPHIE OWES LISA 8.185
```



MULTIPLE PLOTTING

THE FLOW CHART:



THE PROGRAM:

```
10 REM LINE 20 PRINTS A HEADING
20 PRINT "X      Y"
30 LET X=0
40 LET Y=X↑2+2*X+10
50 PRINT X;"     "; Y
60 LET X=X+0.1
70 IF X<=2 THEN 40
80 END
```

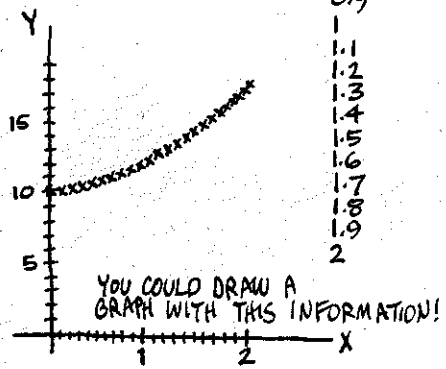
5 SPACES

3 SPACES

RUN

X	Y
0	10
0.1	10.21
0.2	10.44
0.3	10.69
0.4	10.96
0.5	11.25
0.6	11.56
0.7	11.89
0.8	12.24
0.9	12.61
1	13
1.1	13.41
1.2	13.84
1.3	14.29
1.4	14.76
1.5	15.25
1.6	15.76
1.7	16.29
1.8	16.84
1.9	17.41
2	18

GAPS A RESULT OF NOT USING ALL OF BASIC'S FORMATTING ABILITIES!



YOU COULD DRAW A GRAPH WITH THIS INFORMATION!

THE "MULTIPLE PLUG-INS" LOOP IS SO TYPICAL THAT ALL PROGRAMMING LANGUAGES HAVE SPECIAL COMMANDS JUST FOR SUCH REPETITIONS. IN BASIC, IT'S ⟹

FOR NEXT

THIS REPLACES THESE THREE LINES:

```
30 LET X=0
:
60 LET X=X+0.1
70 IF X<=2 THEN 30
```

WITH THESE TWO:

```
30 FOR X=0 TO 2 STEP 0.1
:
60 NEXT X
```

LOWER LIMIT

UPPER LIMIT

THE STATEMENT INITIALLY SETS THE VARIABLE EQUAL TO THE LOWER LIMIT, EXECUTES THE LINES UP TO "NEXT," INCREMENTS THE VARIABLE BY THE AMOUNT "STEP," AND REPEATS THE LOOP UNTIL THE UPPER LIMIT IS EXCEEDED.

A SIMPLE EXAMPLE:

```
10 FOR I = 1 TO 4
20 PRINT I*I
30 NEXT I
40 END
RUN
 1
 4
 9
 16
```

OMITTING "STEP" AUTOMATICALLY MAKES INCREMENT = 1.

NEXT PAGE!

PROBLEMS

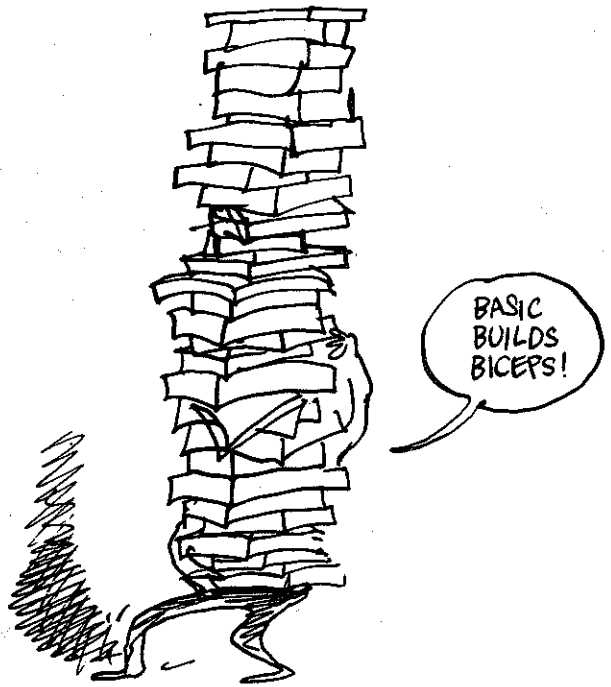
PROBLEMS? WHO HAS PROBLEMS?

1. WHAT DOES THIS PROGRAM DO?

```
10 INPUT N
20 FOR I=1 TO N
30 PRINT I*I
40 NEXT I
50 END
```

2. REWRITE THE "MULTIPLE PLUG-INS" PROGRAM USING THE "FOR NEXT" STATEMENT.
3. WRITE A PROGRAM WHICH ADDS THE INTEGERS (WHOLE NUMBERS) FROM 1 TO 1,000,000. DITTO FROM 1 TO N, FOR ANY N.
4. IN THE FIBONACCI SEQUENCE 0, 1, 1, 2, 3, 5, 8, 13, 21, 34,... EACH NUMBER IS THE SUM OF THE PREVIOUS TWO NUMBERS. WRITE A PROGRAM WHICH GENERATES THIS SEQUENCE.
5. READ ENOUGH OF A BASIC TEXTBOOK TO WRITE A "ROOMMATE RECEIPTS" PROGRAM FOR ANY NUMBER OF ROOMMATES.

THERE ARE PLENTY OF OTHER BASIC FEATURES, ENOUGH TO FILL ENTIRE BOOKS — AND IN FACT TONS OF BOOKS ON BASIC HAVE BEEN PUBLISHED.

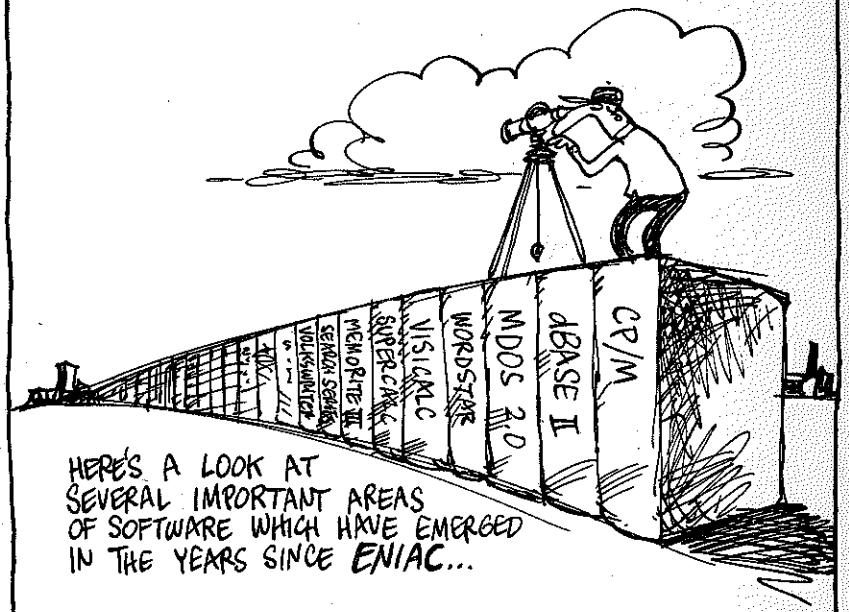


SO.. IF YOU'RE INTERESTED IN DISCOVERING STRING VARIABLES, SUBROUTINES, FUNCTIONS, ARRAYS, NESTED LOOPS, HOW TO DEAL WITH DISKS AND AVOID BUGS, ETC ETC ETC, THEN GO TO YOUR LOCAL LIBRARY OR BOOKSTORE AND GET STARTED !!

AND OTHERWISE?

I'M SORAY... THAT POSSIBILITY IS AGAINST UNIVERSITY POLICY...

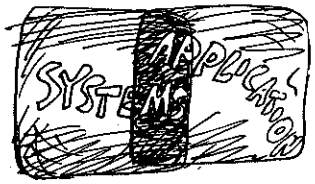
SOFTWARE SURVEY



HERE'S A LOOK AT SEVERAL IMPORTANT AREAS OF SOFTWARE WHICH HAVE EMERGED IN THE YEARS SINCE *ENIAC*...

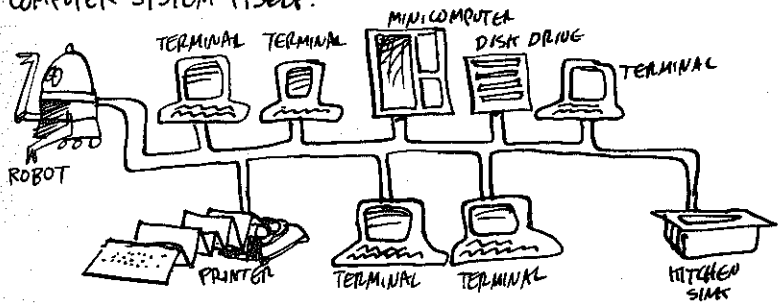
SYSTEMS SOFTWARE

PROGRAMS ARE COMMONLY DIVIDED INTO SYSTEMS SOFTWARE AND APPLICATIONS SOFTWARE.

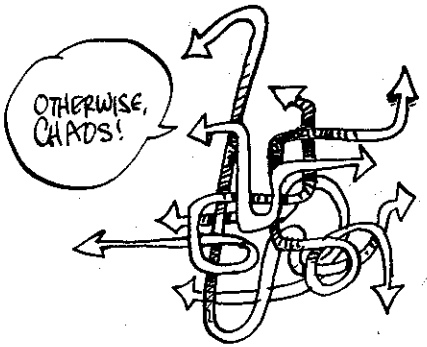


WITH SOME GRAY AREA OF OVERLAP!

APPLICATIONS SOFTWARE DOES "REAL WORLD" JOBS, WHILE SYSTEMS SOFTWARE EXISTS PURELY TO REGULATE THE COMPUTER SYSTEM ITSELF.



A SYSTEM TYPICALLY CONSISTS OF ONE OR MORE INPUT/OUTPUT DEVICES (TERMINALS, PRINTERS, CARD READERS, COMMUNICATIONS PORTS), PROCESSORS, MEMORY UNITS (MAIN AND MASS), AND WHO KNOWS WHAT ELSE. SOMETHING HAS TO COORDINATE IT ALL!



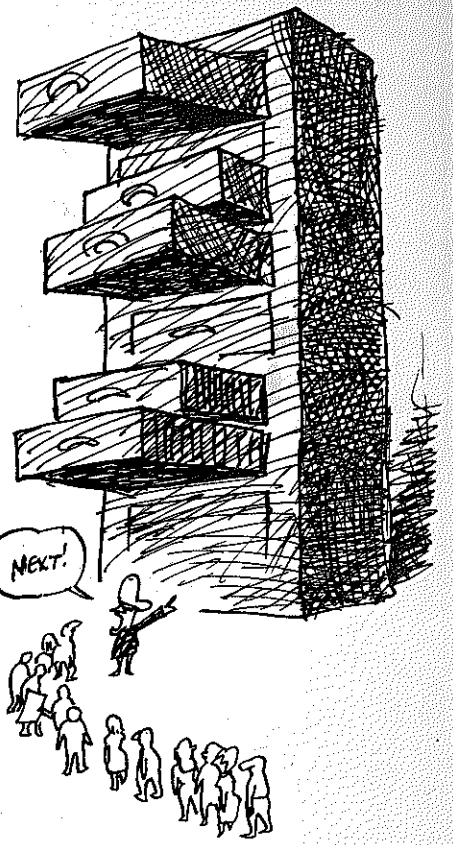
THE PROGRAM THAT DOES IT IS CALLED THE

OPERATING SYSTEM.

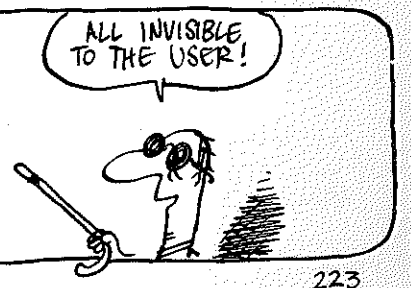
IF YOU THINK OF THE COMPUTER'S CORE AS A GIANT ELECTRONIC FILING CABINET (WITH A CALCULATOR ATTACHED), THEN THE OPERATING SYSTEM

- ☆ CREATES THE STRUCTURE OF THE FILES
- ☆ MANAGES MEMORY SO THAT DIFFERENT FILES DON'T BUMP INTO EACH OTHER
- ☆ REGULATES ACCESS TO THE FILES AND THE MOVEMENT OF INFORMATION TO AND FROM OTHER PARTS OF THE SYSTEM...

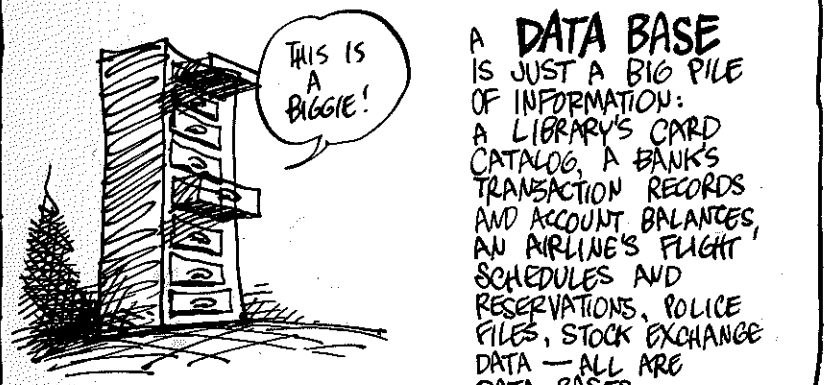
ETC!



BESIDES THE OPERATING SYSTEM, SYSTEM SOFTWARE INCLUDES OTHER PROGRAMS "IN THE SYSTEM," SUCH AS LOADERS (WHICH LOAD PROGRAMS INTO MEMORY) AND COMPILERS (WHICH TRANSLATE HIGHER-LEVEL LANGUAGE INTO MACHINE CODE).



DATA BASE MANAGEMENT



A **DATA BASE** IS JUST A BIG PILE OF INFORMATION: A LIBRARY'S CARD CATALOG, A BANK'S TRANSACTION RECORDS AND ACCOUNT BALANCES, AN AIRLINE'S FLIGHT SCHEDULES AND RESERVATIONS, POLICE FILES, STOCK EXCHANGE DATA — ALL ARE DATA BASES.

A DATA BASE MANAGEMENT PROGRAM ORGANIZES, UPDATES, AND PROVIDES ACCESS TO THE DATA BASE.

IN THE CASE OF AN AIRLINE, FOR EXAMPLE, THE COMPUTER HAS TO BOOK RESERVATIONS, ASSIGN SEATS, ERASE RESERVATIONS WHEN THE CUSTOMER CANCELS, MAKE REASSIGNMENTS IF A FLIGHT IS CANCELED, PRINT THE TICKETS, AND PROVIDE ALL THE FLIGHT INFORMATION TO TRAVEL AGENTS — WORLDWIDE!!

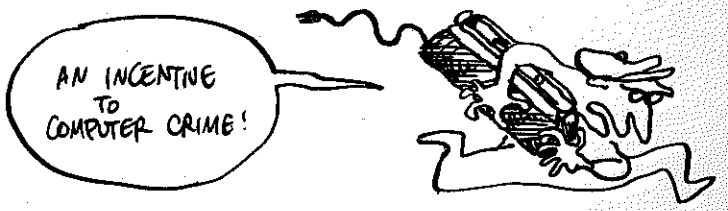
WORD PROCESSING

A "PERSONAL" USE FOR COMPUTERS...

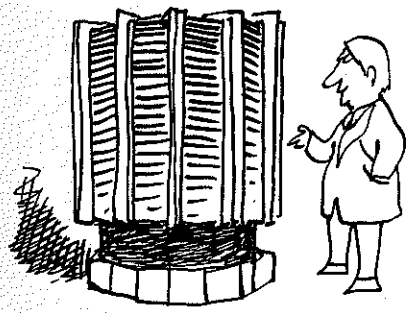
WORD PROCESSING SOFTWARE ALLOWS YOU TO WRITE, EDIT, AND FORMAT TEXT — ALL FROM THE SAME KEYBOARD. YOU CAN GO FROM FIRST TO FINAL DRAFT ELECTRONICALLY, BEFORE EVER PRINTING A WORD.

THERE ARE ALSO PROGRAMS TO CORRECT SPELLING — AND EVEN TO FIX SYNTAX AND GRAMMAR. SOON ILLITERATES WILL BE CREATING MASTERPIECES!

A SMALL COMPUTER WITH WORD PROCESSING CAN BE QUITE INEXPENSIVE... THE CATCH IS THAT A "LETTER QUALITY" PRINTER CAN COST TEN TIMES THE PRICE OF A TYPEWRITER!



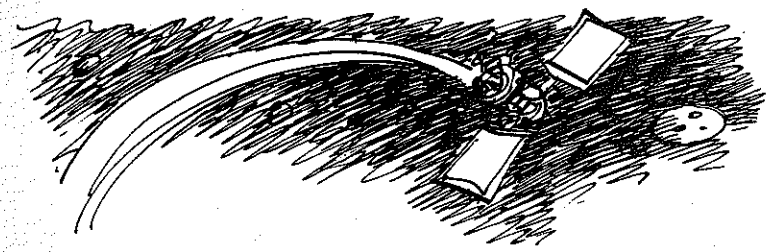
SCIENCE



CRAY-1 COMPUTER, CAPABLE OF 100 MILLION OPERATIONS PER SECOND!!

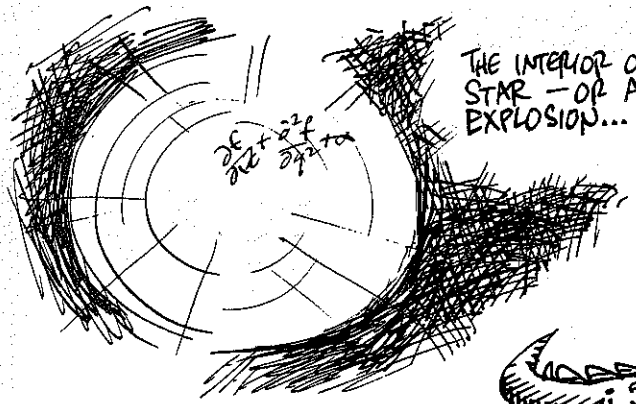
SCIENCE DEPENDS ON MATHEMATICS AND COMPUTERS ARE SUPER MATH MACHINES. THE FASTEST, MOST POWERFUL COMPUTERS ARE MAINLY APPLIED TO SCIENTIFIC PROBLEMS.

THESE "SUPERCOMPUTERS" EXCEL AT **SIMULATION.** THE IDEA BEHIND SIMULATION IS TO FEED THE COMPUTER THE EQUATIONS GOVERNING A PHYSICAL SYSTEM AND THEN MATHEMATICALLY "MOVE" THE SYSTEM ACCORDING TO THOSE LAWS.



TAME SPACE TRAVEL: A COMPUTER CAN GUIDE A CRAFT TO THE MOON, BECAUSE IT CAN INTERNALLY SIMULATE THE ENTIRE FLIGHT!!

COMPUTERS CAN SIMULATE:



THE INTERIOR OF A STAR — OR A NUCLEAR EXPLOSION...

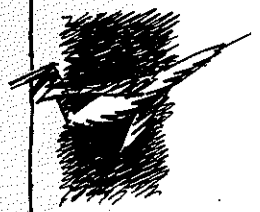
THE EVOLUTION OF AN ECOSYSTEM...

A LENS

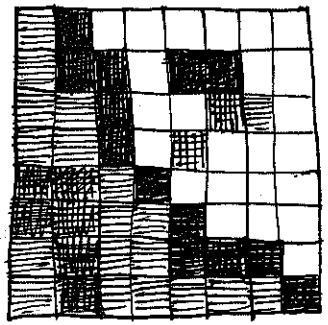
CLIMATE (ALTHOUGH EVEN THE FASTEST COMPUTER ISN'T FAST ENOUGH TO PREDICT THE WEATHER).

I'LL TELL YOU TOMORROW'S WEATHER NEXT WEEK!

GRAPHICS



FROM THE SIMPLEST "PONG" SCREEN TO THE MOST SOPHISTICATED FLIGHT SIMULATOR, THE IDEA IS THE SAME:



DIVIDE THE SCREEN AREA INTO A LARGE NUMBER OF TINY RECTANGLES ("PIXELS") AND ASSIGN EACH ONE A COLOR AND BRIGHTNESS.

THAT'S WHY COMPUTER PICTURES HAVE CORNERS!

BUT THERE ARE ALSO ALGORITHMS FOR SMOOTHING CORNERS!

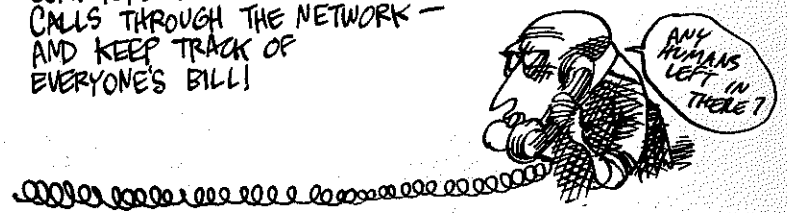
UNFORTUNATELY, IT TAKES A LOT OF COMPUTER POWER TO DO FANCY GRAPHICS. SMALL COMPUTERS MOSTLY DO THINGS LIKE MAKE PIE CHARTS...

COMMUNICATION

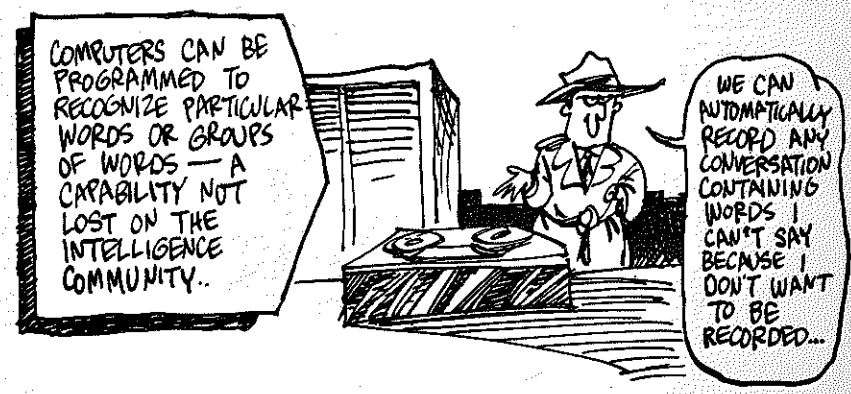
THE BIGGEST COMPUTER SYSTEM OUTSIDE GOVERNMENT BELONGS TO THE **TELEPHONE COMPANY.**

A VOICE (OR ANY OTHER SIGNAL) CAN BE DIGITALLY ENCODED, TRANSMITTED, AND DECODED.

COMPUTERS ALSO CONTROL THE ROUTING AND SWITCHING OF CALLS THROUGH THE NETWORK — AND KEEP TRACK OF EVERYONE'S BILL!



COMPUTERS CAN BE PROGRAMMED TO RECOGNIZE PARTICULAR WORDS OR GROUPS OF WORDS — A CAPABILITY NOT LOST ON THE INTELLIGENCE COMMUNITY..

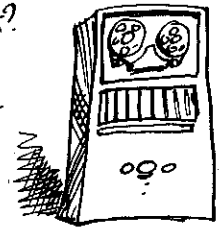


ARTIFICIAL INTELLIGENCE

DESPITE THEIR INCREDIBLE SPEED AND ACCURACY, COMPUTERS ARE LOUSY AT PATTERN RECOGNITION, ANALYSIS, HUNCH-PLAYING, AND UNDERSTANDING HUMAN LANGUAGE!

CAN A MACHINE BE PROGRAMMED TO THINK?

ER... WELL... UM... AH... LET ME SEE...

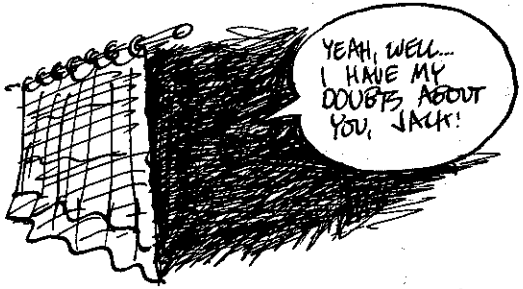


ACTUALLY, WE KNOW VERY LITTLE ABOUT HOW THINKING WORKS...

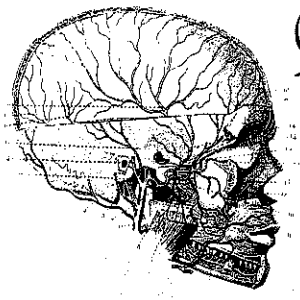
SO A BETTER QUESTION IS: HOW CAN YOU TELL IF A MACHINE IS THINKING?

ALAN TURING SUGGESTED THIS TEST: SUPPOSE YOU COULD COMMUNICATE WITH SOMETHING, OR SOMEONE, CONCEALED FROM VIEW. IF, ON THE BASIS OF THE CONVERSATION, YOU COULDN'T SAY WHETHER IT WAS MACHINE OR HUMAN, YOU WOULD HAVE TO SAY IT WAS THINKING!

IT'S A MACHINE!



I PERSONALLY DISLIKE THIS CRITERION, ON THE GROUNDS THAT A SIMULATION ISN'T THE REAL THING...

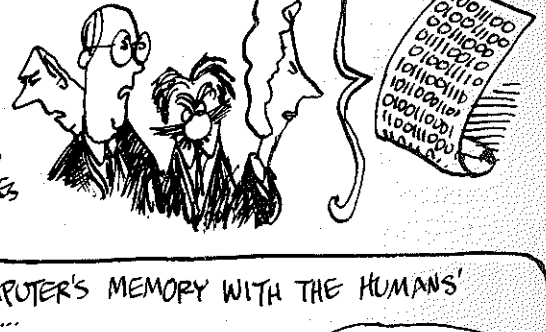


ONWARD!

THIS PHILOSOPHICAL MUDDLE HASN'T STOPPED PEOPLE FROM TRYING TO MAKE MACHINES THINK. THEY'VE HAD SOME SUCCESS WITH SO CALLED EXPERT SYSTEMS, WHICH MIMIC HUMAN EXPERTS IN VARIOUS FIELDS.

HOW DO YOU CREATE AN EXPERT SYSTEM?

FIRST, INTERVIEW A BUNCH OF EXPERTS — GEOLOGISTS, FOR EXAMPLE — AND FORCE THEM TO SPELL OUT THE ALGORITHMS BEHIND THEIR SKILLS, HUNCHES AND BRAINSTORMS.



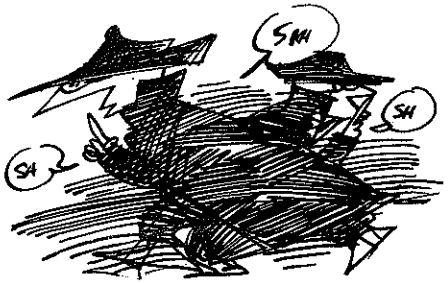
THEN LOAD THE COMPUTER'S MEMORY WITH THE HUMANS' KNOWLEDGE BASE... AND THE RESULT IS (SOMETIMES) A PROGRAM WHICH CAN OUTPERFORM ANY HUMAN!!

IF WE'RE SO SMART, WHY DID WE LET THIS HAPPEN?

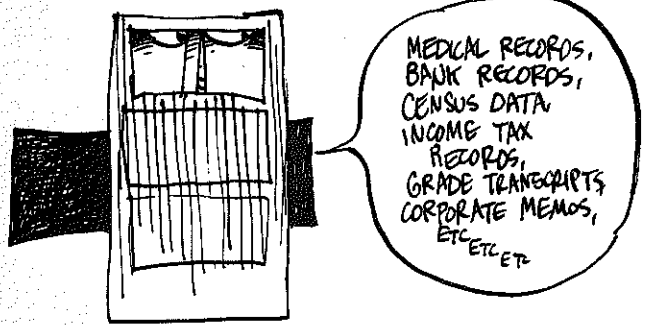
CRYPTOGRAPHY

SHHH!

THERE ARE STANDARD CODES LIKE ASCII (P. 128) FOR CONVERTING WRITTEN TEXT INTO BINARY... BUT WHAT ABOUT USING COMPUTERS FOR **SECRET** CODES??



SECRET CODES USED TO BE STRICTLY MILITARY AND SPY STUFF, BUT NOW MORE AND MORE SENSITIVE INFORMATION IS STORED IN COMPUTER SYSTEMS:



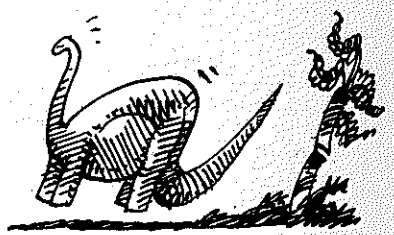
SCRAMBLING DATA HAS BECOME AN IMPORTANT WAY OF PROTECTING PRIVACY!!

ORDINARILY, INFORMATION IS STORED AS A BINARY STRING ANY COMPUTER CAN READ: THE ***PLAINTEXT***, IN CRYPTOGRAPHIC JARGON. TO ENCRYPT IT YOU APPLY SOME ALGORITHM **S**, WHICH CONVERTS IT TO A SCRAMBLED MESSAGE CALLED THE ***CYPHERTEXT***.

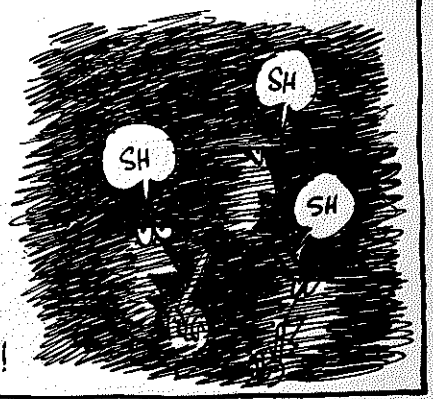
PLAINTEXT —S→ CYPHERTEXT

THEORETICALLY, IT'S IMPOSSIBLE TO RECONSTRUCT THE PLAINTEXT FROM THE CYPHERTEXT WITHOUT KNOWING SOMETHING ABOUT S ... HOWEVER, A POTENTIAL CODE-BREAKER COULD PUT A COMPUTER TO WORK SEARCHING FOR S.

TO BE SECURE, S HAS TO BE SO COMPLICATED THAT EVEN THE FASTEST COMPUTER WOULD TAKE, SAY, A FEW MILLION YEARS TO FIGURE IT OUT!



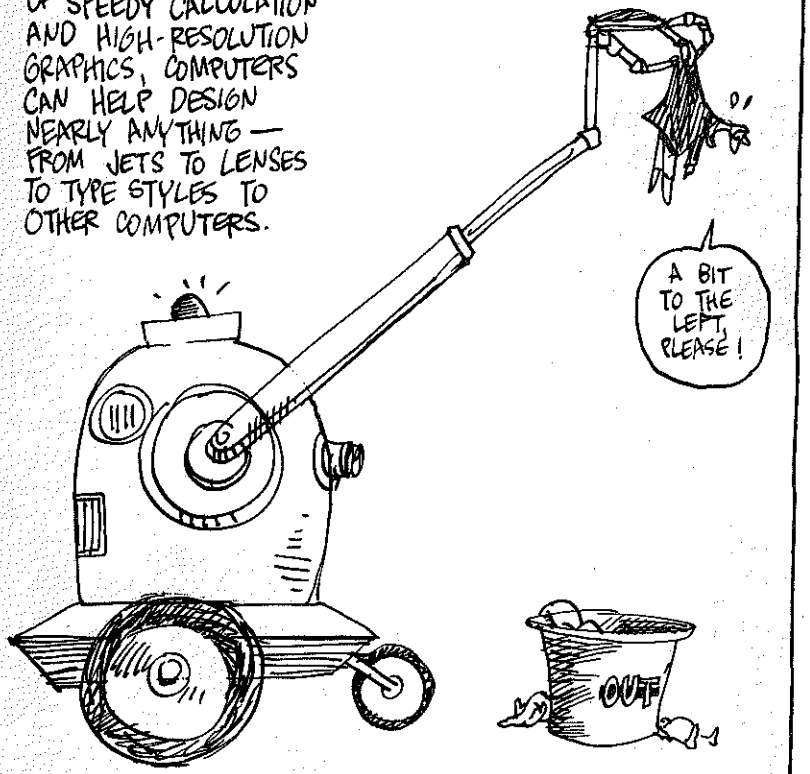
RECENTLY, THE NATIONAL BUREAU OF STANDARDS APPROVED A FAMILY OF ALGORITHMS AS A DATA ENCRYPTION STANDARD FOR THE NATION. SEVERAL SCIENTISTS SUSPECT THAT THIS STANDARD IS JUST COMPLEX ENOUGH TO STYMIE ORDINARY COMPUTERS, BUT NOT TOO TOUGH FOR THE *NINE* ACRES OF COMPUTERS OF THE NATIONAL SECURITY AGENCY!



CAD/CAM

: COMPUTER-AIDED DESIGN /
: COMPUTER-AIDED MANUFACTURE

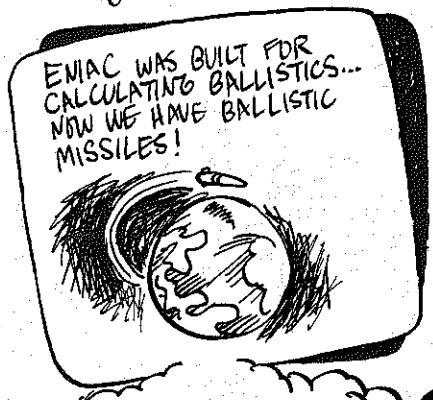
USING A COMBINATION OF SPEEDY CALCULATION AND HIGH-RESOLUTION GRAPHICS, COMPUTERS CAN HELP DESIGN NEARLY ANYTHING — FROM JETS TO LENSES TO TYPE STYLES TO OTHER COMPUTERS.



THEN THEY CAN GO ON TO CONTROL AUTOMATIC MANUFACTURING PROCESSES AS WELL. YES, ROBOTS ARE ALREADY HERE!!

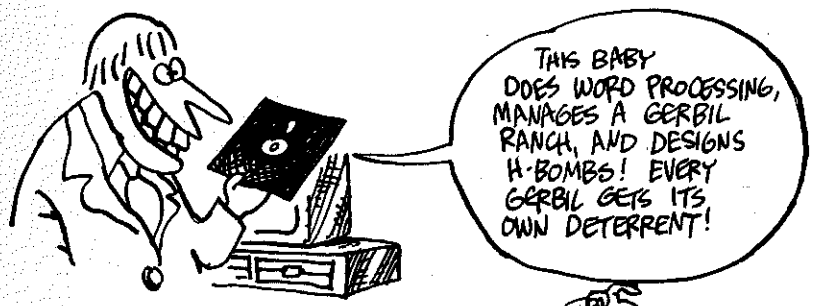
WAR

THE MILITARY CAN USE JUST ABOUT EVERY TYPE OF SOFTWARE WE'VE MENTIONED — AND THEN SOME!

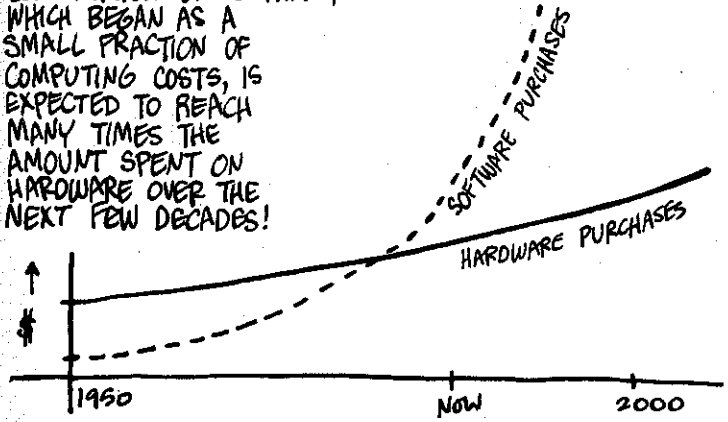


...NOT TO MENTION DATA PROCESSING AND CRYPTOGRAPHY... SO GREAT IS THE DEFENSE DEPARTMENT'S SOFTWARE NEED THAT THEY HAVE THEIR OWN PROGRAMMING LANGUAGE: ***ADA***, NAMED AFTER THE UNFORTUNATE LADY LOVELACE.

THIS LITTLE SURVEY ONLY BEGINS TO SUGGEST THE RANGE OF SOFTWARE CURRENTLY AVAILABLE. EVERY DAY THERE'S MORE... SOME PROGRAMS MOVE INTO NEW AREAS, WHILE OTHERS INTEGRATE EXISTING ROUTINES INTO NEW, MORE POWERFUL PACKAGES.



IF YOU'RE LOOKING FOR OPPORTUNITY IN THE COMPUTER BUSINESS, CONSIDER THIS: THE TOTAL CONSUMPTION OF SOFTWARE, WHICH BEGAN AS A SMALL FRACTION OF COMPUTING COSTS, IS EXPECTED TO REACH MANY TIMES THE AMOUNT SPENT ON HARDWARE OVER THE NEXT FEW DECADES!



IN CONCLUSION,

A FEW WORDS ABOUT THIS FAMILIAR SENTENCE:

(WHICH IS WHAT COMPUTER SCIENTISTS SAY WHEN THEY WANT TO BE REASSURING...)

TECHNICALLY, IT'S TRUE, IN THE SENSE THAT SOFTWARE CONTROLS COMPUTERS, AND PEOPLE WRITE SOFTWARE...

BUT WHO CONTROLS PEOPLE?!!

FOR EXAMPLE, SUPPOSE A NATION'S STRATEGIC PLANNERS DECIDED TO PROGRAM THEIR COMPUTERS TO ORDER A MISSILE ATTACK AUTOMATICALLY "ON WARNING." CONSIDERING THAT U.S. DEFENSE COMPUTERS SOUND SEVERAL FALSE ALARMS A YEAR, IS THIS REASSURING??

I WAS ONLY FOLLOWING INSTRUCTIONS ...

MY CALCULATOR SAYS $2^{16} = 65,536.001$ (REALLY!)

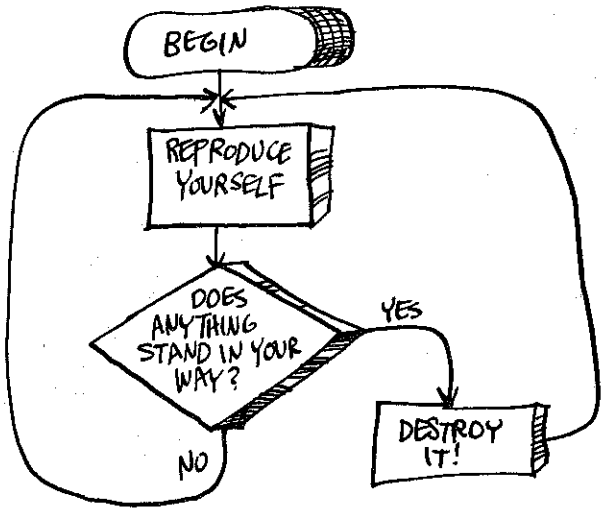
ANOTHER PROBLEM IS THAT ALGORITHMS DON'T ALWAYS DO EXACTLY WHAT THEY ARE SUPPOSED TO.

LARGE SOFTWARE SYSTEMS ARE WRITTEN BY TEAMS OF PROGRAMMERS. LIKE THE ELEPHANT, NO ONE UNDERSTANDS THE WHOLE THING!

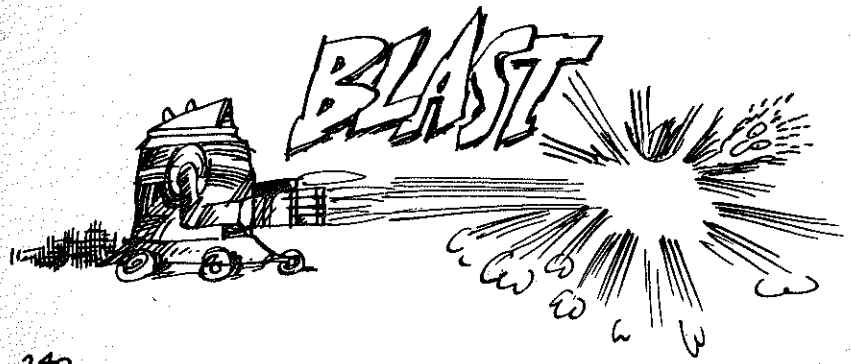
COMPUTERS ROUTINELY DO BIZARRE AND UNEXPECTED THINGS, ESPECIALLY WHEN RUNNING NEW, UNTESTED SOFTWARE!

I'M GETTING TO BE A TIRED METAPHOR!

FINALLY, CONSIDER THIS OMINOUS ALGORITHM:



WHILE NO COMPUTER IS INTELLIGENT, MOBILE, OR WELL EQUIPPED ENOUGH — YET — TO EXECUTE THESE INSTRUCTIONS, SUCH A MACHINE REMAINS A THEORETICAL POSSIBILITY. THIS PROGRAM WOULD MAKE IT SOMETHING VERY MUCH LIKE A COMPETING LIFE FORM!!!



AND IF YOU THINK THAT BECAUSE "IT'S ONLY A MACHINE," YOU CAN ALWAYS TURN IT OFF, PONDER THE WORDS OF NORBERT WIENER, A SCIENTIST WHO THOUGHT DEEPLY ABOUT THESE THINGS:

"TO TURN A MACHINE OFF EFFECTIVELY, WE MUST BE IN POSSESSION OF INFORMATION AS TO WHETHER THE DANGER POINT HAS COME. THE MERE FACT THAT WE HAVE MADE THE MACHINE DOES NOT GUARANTEE THAT WE SHALL HAVE THE PROPER INFORMATION TO DO THIS.... THE VERY SPEED OF... MODERN DIGITAL MACHINES STANDS IN THE WAY OF OUR ABILITY TO PERCEIVE AND THINK THROUGH THE THE INDICATIONS OF DANGER." *

* CYBERNETICS, SECOND EDITION, P. 175

SO WELCOME TO THE INFORMATION AGE, AND HAPPY COMPUTING !!

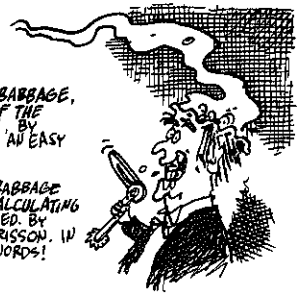
SOME FURTHER READING:

MEDIEVAL AND EARLY MODERN SCIENCE BY A.C. CROMBIE. TELLS HOW ISLAMIC SCIENCE CAME TO EUROPE.

THE MAKING OF THE MICRO BY C. EVANS. NICE DIAGRAMS OF OLD ADDING MACHINES

HISTORY OF MATHEMATICS BY A. GITTLEMAN. DON'T MISS THE STORY OF NAPIER'S "PSYCHIC" CHICKEN!

THE COMPUTER FROM PASCAL TO VON NEUMANN BY H. GOLDSTINE. THE DEFINITIVE ACCOUNT OF ENIAC.



CHARLES BABBAGE, FATHER OF THE COMPUTER BY D. KALACY. AN EASY READ.

CHARLES BABBAGE AND HIS CALCULATING ENGINES, ED. BY P. & E. MORRISSON. IN HIS OWN WORDS!

SYMBOLIC LOGIC AND THE GAME OF LOGIC BY LEWIS CARROLL. MILLIONS OF SILLY SYLLOGISMS

THE MATHEMATICAL THEORY OF COMMUNICATION BY C. SHANNON. CONTAINS TWO PARTS, ONE WITH AND ONE WITHOUT MATH

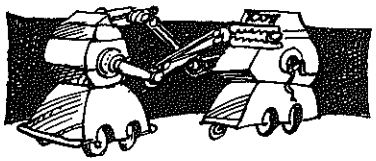
CYBERNETICS, 2ND EDITION, BY N. WEINER. THE THEORY OF AUTOMATIC CONTROL

UNDERSTANDING DIGITAL ELECTRONICS, BY D. McWHORTER. BOOLEAN CIRCUITS

UNDERSTANDING DIGITAL COMPUTERS, BY P. MIMS. A PERSONAL FAVORITE, BUT LOOK OUT FOR MISPRINTS!

INTRODUCTION TO MICROCOMPUTERS, BY A. OSBORNE (4 VOLUMES). VERY DETAILED

UNDERSTANDING COMPUTER SCIENCE, BY R.S. WALKER. MORE ADVANCED TOPICS



ILLUSTRATING BASIC BY D. ALCOCK. A QUICK COURSE, USING QUASI-CARTOONS

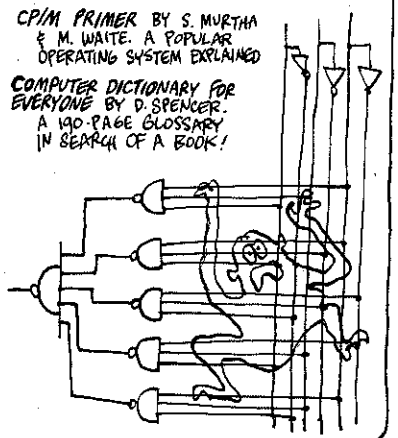
USING BASIC, BY R. DIDDAY & R. PAGE. A GENTLER, BUT WORDIER, APPROACH

PASCAL PRIMER BY D. FOX & M. WAITE. IT HELPS TO KNOW BASIC BEFORE READING THIS

FORTRAN COLORING BOOK BY R. KAUFMAN. WITTY, BORDERING ON CORNY

CP/M PRIMER BY S. MURTHA & M. WAITE. A POPULAR OPERATING SYSTEM EXPLAINED

COMPUTER DICTIONARY FOR EVERYONE BY D. SPENCER. A 190-PAGE GLOSSARY IN SEARCH OF A BOOK!



INDEX