CHAPTER 2

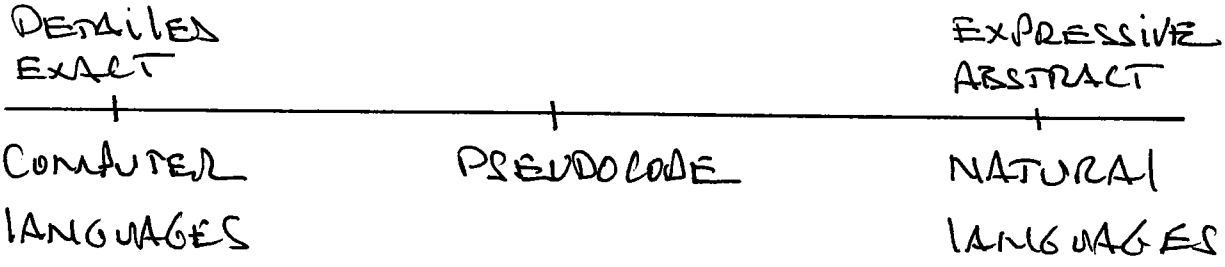
ALGORITHM DESIGN

HOW SHOULD WE PRESENT ALGORITHMS ?

NATURAL LANGUAGES LIKE ENGLISH WHICH ARE RICH IN INTERPRETATION AND MEANING ARE NOT IDEAL FOR THIS PURPOSE. WE NEED A MORE PRECISE NOTATION TO PRESENT ALGORITHMS, NOT SUBJECT TO INTERPRETATION.

AT THE OTHER EXTREME FORMAL PROGRAMMING LANGUAGES (C/C++, PASCAL, ADA, ...) ARE VERY INFLEXIBLE IN THEIR SYNTAX, AND REQUIRE MUCH ATTENTION TO DETAIL; DETAILS WHICH MAY BE IRRELEVANT IN INITIAL DESIGN PHASES.

INSTEAD WE USE AN INFORMAL AND FLEXIBLE LANGUAGE CALLED PSEUDOCODE WHICH USES ENGLISH LANGUAGE CONSTRUCTS AND CONVENTIONS MODELED TO LOOK LIKE COMMANDS AVAILABLE IN MOST COMPUTER LANGUAGES.



PSEUDOCODE IS A LANGUAGE IN WHICH ALL NECESSARY OPERATIONS CAN BE EXPRESSED, BUT WHICH IS INDEPENDENT OF ANY COMPUTER LANGUAGE, OR COMPUTER FOR THAT MATTER.

SEQUENTIAL OPERATIONS

ARE OF THREE BASIC KINDS: COMPUTATION, INPUT, AND OUTPUT.

THE OPERATION OF PERFORMING A CALCULATION AND STORING THE RESULT IS WRITTEN

SET "VARIABLE" TO "EXPRESSION"

(OR "variable" ← "expression")

THIS INSTRUCTION TELLS THE COMPUTING AGENT TO EVALUATE THE ARITHMETIC EXPRESSION "EXPRESSION" AND PLACE ITS VALUE IN "VARIABLE".

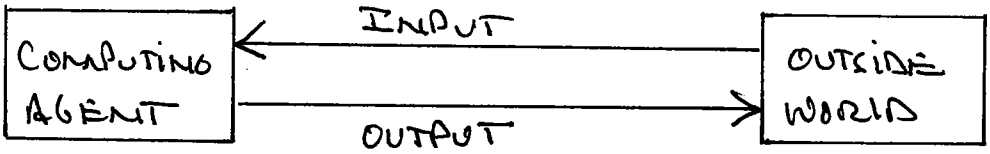
EX. SET C_i TO $a_i + b_i + \text{carry}$.

WE USUALLY ASSUME OUR COMPUTING AGENT IS AS CAPABLE AS ANY POCKET CALCULATOR.

EX. SET x TO $\dfrac{-b + \sqrt{b^2 - 4ac}}{2a}$.

INPUT AND OUTPUT

OPERATIONS ALLOW THE COMPUTING AGENT TO RECIEVE DATA VALUES FROM, AND SEND RESULTS TO THE OUTSIDE WORLD (I.E. THE USER.)



GET VALUES FOR VARIABLE_1, VARIABLE_2 . . .
PRINT VALUES OF VARIABLE_1, VARIABLE_2, . . .

PRINTED TEXT WILL BE PLACED IN SINGLE QUOTES.

EX. PRINT THE MESSAGE 'ERROR: DIVISION BY ZERO'

AN ALGORITHM WHICH USES ONLY SEQUENTIAL OPERATIONS IS SOMETIMES CALLED A STRAIGHT LINE ALGORITHM

EX. GIVEN REAL NUMBERS a, b, c, d, FIND THEIR AVERAGE, (a+b+c+d)/4.

1.) GET VALUES FOR a, b, c, d.
2.) SET SUM TO a+b+c+d
3.) SET AVERAGE TO SUM/4
4.) PRINT VALUE OF AVERAGE.

CONDITIONAL OPERATIONS

ARE WRITTEN AS

```
if "CONDITION"
    [DO SOMETHING
        :
        :
```

MORE GENERALLY WE MAY WRITE

```
if "CONDITION"
    [DO SOMETHING
        :
        :

ELSE
    [DO SOMETHING
        :
        :
```

Ex.

1.) GET VALUES FOR a, b
2.) if b = 0
3.) PRINT 'ERROR: DIVISION BY ZERO'
4.) ELSE
5.) SET QUOTIENT TO a/b
6.) PRINT VALUE OF QUOTIENT
7.) STOP

ITERATIVE OPERATIONS

i.e. looping instructions, are written

```
    REPEAT i TO j UNTIL "CONDITION"
i.)     DO SOMETHING
          :
j.)     DO SOMETHING
```

HERE "CONDITION" is ANY STATEMENT which HAS A TRUTH VALUE (i.e. 'TRUE' or 'FALSE'). THIS is CALLED THE <u>LOOP TERMINATION CONDITION</u> STEPS i TO j ARE CALLED THE <u>LOOP BODY</u>. THE LOOP BODY is REPEATED UNTIL "CONDITION" TESTS FALSE

EX

```
1.) REPEAT 2 TO 9 UNTIL RESPONSE is 'NO'
2.)     GET VALUES FOR a, b FROM USER
3.)     if  b=0
4.)         PRINT 'ERROR: DIVISION BY ZERO'
5.)     ELSE
6.)         SET q TO a/b
7.)         PRINT VALUE OF q
8.)     PRINT 'DO YOU WISH TO CONTINUE?'
9.)     GET RESPONSE FROM USER.
10.) STOP
```

AN ALTERNATIVE WAY TO GIVE LOOPING INSTRUCTIONS IS TO USE A <u>WHILE LOOP</u>

```
    while "CONDITION" DO i TO j
i.)     DO SOMETHING
          :
          :
j.)     DO SOMETHING.
```

HERE THE LOOP BODY IS REPEATED AS LONG AS "CONDITION" IS TRUE. IN THIS CASE "CONDITION" IS CALLED THE <u>LOOP REPITITION CONDITION</u>.

NOTE A REPEAT LOOP MUST EXECUTE AT LEAST ONCE; A WHILE LOOP MAY EXECUTE 0 TIMES.

<u>EX.</u>

```
1.) SET RESPONSE TO 'YES'
2.) WHILE RESPONSE IS YES DO
3.)     GET VALUES FOR a, b
4.)     IF b = 0
 !        !
 !        !
10.)    GET RESPONSE
11.) STOP
```

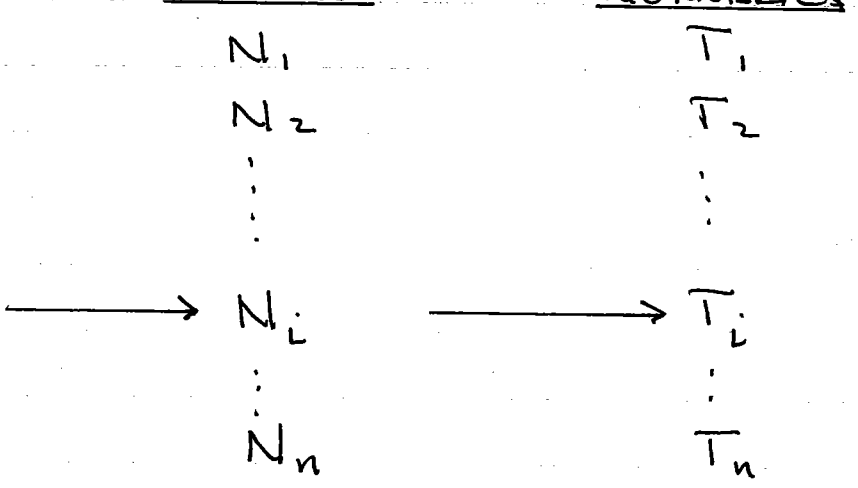
- DO EX WITH for LOOP.

IT HAS BEEN SHOWN THAT ONE CAN REPRESENT ANY VALID ALGORITHM WITH THE OPERATIONS SO FAR DESCRIBED.

SEE OPERATIONS IN FIGURE 2.6, P. 42

ALGORITHM EXAMPLES

CONSIDER THE PROBLEM OF LOOKING UP A NAME IN A TELEPHONE DIRECTORY.



WE ASSUME THAT THE NAMES ARE IN NO PARTICULAR ORDER (i.e. NOT NECESSARILY ALPHABETICAL) SO THAT WE MUST LOOK AT EACH NAME IN SUCCESION.

IF WE FIND A NAME N_i WHICH MATCHES, WE SHOULD RETURN THE NUMBER T_i. IF NO N_i IN A MATCH WE SHOULD PRINT A MESSAGE TO THAT EFFECT

SEQUENTIAL SEARCH

GIVEN:

$n \geq 1$, $N_1, \ldots, N_n$, $T_1, \ldots, T_n$, AND NAME (THE NAME TO SEARCH FOR.)

FIND:

THE NUMBER T_i FOR WHICH NAME $= N_i$. IF NONE EXISTS, PRINT A MESSAGE TO THE USER.

1.) SET i TO 1
2.) SET FOUND TO FALSE
3.) REPEAT 4 TO 8 UNTIL $i > n$ OR FOUND
4.) IF N_i = NAME
5.) SET FOUND TO TRUE
6.) PRINT T_i
7.) ELSE
8.) SET i TO $i+1$
9.) IF NOT FOUND
10.) PRINT 'SORRY, NAME NOT FOUND'
11.) STOP

NOTE THAT THE LOOP TERMINATION CONDITION $i > n$ OR FOUND HAS TWO PARTS.