Computers for the Arts

DICK HIGGINS

ABYSS Publications

Computers for the Arts

DICK HIGGINS

Computers are like most tools—deaf, blind and incredibly stupid. So stupid, in fact, that they cannot imagine how to make a mistake once they are programmed to do what is expected of them. This makes them different from other tools. Imagine a hammer which, once programmed to build a table, could do so on its own, without the possibility of damaging or splitting the wood. It would leave the carpenter free to concentrate on the design of his table with no worry about the difficulties of execution. The role which computers can play is analogous to this.

Computers when used for the arts are doing what they are not normally designed to do. Their main use is economic – in science and business. However, their uses are sufficiently versatile to justify looking into a number of the special techniques for the solution of creative problems.

We do not communicate with a computer by telling it verbally what to do. Instead we provide a structure on the basis of which it processes whatever data we provide. The artificial languages in which these programs are written vary, and some are less comprehensible to the mathematically unskilled than others. In my own experience I have found that, in spite of my lack of formal training in logic or mathematics, it is easier to pick up the basics of communicating with a computer in one of the commonest of these artificial languages, Fortran, than to try to explain my intentions to an engineer who lacks my motivations.

Suppose my project is a verbal one, resulting in a listing of words which will resemble verse. There are two parts to achieving this: (1) my materials (verbs, nouns, or whatever I happen to be using), which, for the machine, are simply

data; (2) the method of treating these materials – the program itself. Within the program I will have to define a number of variables, one of which, for example, might represent the sequential order of each line, another the number of units (words) per line. If I wish this number to vary, I will be able to calculate a value of a third variable to cause this to happen.

To compose such a program, I will have to understand the logic of the process involved, what calculations take place in what order, etc. If I wish to have five words per line, I must decide that "word(1)" represents the first word of the "word(2)" represents the second, and so on. "Line" would therefore represent a set of five values for "word," and would be calculated (and written out by the machine) one fifth as often. The calculation to determine the value of "word" would be logically "nested" inside the calculation to determine the value of "line."

This is, of course, paper work, and is independent of the physical nature of the computers. The artist seldom sees the computer, just as an author, traditionally, seldom sees the printing press that produces his book. The physical operation of a computer is done by an "operator," who may or may not know anything about the program he is feeding into the machine. It is the programmer who prepares the machine to function, just as the musicians and recording engineers have taken care of the necessary artistic and technical questions by the time one places a phonograph record in a turntable.

For many artistic applications the computer system can make use of "analogue devices." While the output of a digital computer consists of a printout on long sheets of paper or magnetic tape containing data, the analogue device can interpret this data in other ways. Once interpreted it can be fed back into a digital computer for further analysis. A drawback of analogue devices is that they are not always available in the main computer systems, and the more specialized machines are extremely expensive. There are times when the use of special equipment is vital. For example, a composer might connect a microphone to some form of analogue device which analyzes all the sounds present when he says "hello"; he might also have the device analyze the sounds of a string quartet. He could then feed the resulting analysis through another analogue device (a sound synthesizer), and obtain a tape or broadcast over a monitoring system of the string quartet saying "hello." Though this is so simple that one could find other, less expensive, electronic means of doing it, in more complex versions of this process it is possible that only analogue devices could solve the composer's problems.

Another interesting analogue device can scan any body of visual material, then distort it systematically along any axis, simple or curved. For example, a line drawing can be scanned, then systematically rotated to become a 45° angular distortion of its original self, and then a straight line. Such devices are used in the design of aircraft parts. But similiar analyses can be made from photographs and projected back according to the desired distortions onto a screen and rephotographed by an animation camera for use in inexpensive cartoons for television, or for special typographic effects.

Digital computers can absorb information only from special tapes which have been encoded with punches, from punched cards, or from magnetic tapes with information from previous computer runs. Usually, in communicating with them, one starts with what is called a "dimension" statement. This tells the machine how to check that it has absorbed all the required information and, in its own "memory mechanism" how much space to allot to the required program. Next it must be told what exceptions there will be to its usual methods. For example, in Fortran it is a convention that variables whose initial letters run from "I" through "N" represent integers, while all the others represent "real" numbers. So, if we are anxious (in the verbal project described) to use 5 units called "WORD" to a line, we must either include the statement in the program "INTEGER WORD," which causes WORD to stand for an integer, or rename the variable to be initialed like an integer, for example – "IWORD" so that the machine will know it is calculating word (1), word(2), etc, and not word (1.85), word (2.6), etc.

Next we ask the computer to read the data we are going to work with, indicating the format – e.g., the card, tapes, or both, etc. The last preliminary step is to ask the computer to write out the data and program that has been fed in, for checking.

Because the ending portions of the programs are similar to the starting procedures – they are both mechanical – I will mention them before returning to the main body of the programs. There is the WRITE statement, which usually has a form like this:

```
50  WRITE (6,102) (LINE(J), J=1, 5)
```

The "50" is simply the number of the statement in the program. The WRITE transfers the calculated information to the output. The "6" indicates an output device number which translates the binary information stored by the machine into our normal decimal and letter system. The "102" indicates that the format of the written statement is given in statement 102. The "LINE(K), J=1, 5" tells the machine what it is to write (LINE), but that LINE is actually a set containing five numbers, each indexed by a value of J that runs from one to five.

The format is defined by equally mechanical conventions. The FORMAT statement can appear anywhere in the program, so long as it has a statement number, but many programmers like to put the FORMAT statement after the WRITE statement it refers to, or to put them all in one place in the program in order to be able to find them easily and quickly. The format referred to in the sample WRITE statement might look something like this

```
102  FORMAT (10X,5A5)
```

if it were intended to produce not a four-word line, as in the illustration *Hank and Mary*, but a five-word one. The FORMAT statement is one of the most difficult things in Fortran to work out, because it is involved in the conception of "Conversion" which indicates how the binary information the machine has now calculated is to be converted into usable data. The "5A5" means that there are 5 units when LINE is written (see the WRITE statement), each of which has 5 letters. The "A" indicates "A-type conversion," which, in the Fortran language, indicates a body of Alphameric (alphabet or numbers) being fed into the machine, indexed, and, then fed out in exactly the same form the data was fed in – that is, the letter will not be reversed, and no cryptographic codes will be used to disguise them. "H" conversion in the FORMAT statement causes the letters (blanks count as characters just as numbers and letters do) following the "H" to be printed out as written in the FORMAT statement. This is important when one wants titles, but note the more significant use in *Proposition No. 2 for Emmett Williams* (see p. 000). "X" conversion simply introduces however many blanks are desired. When numerical outputs are wanted, there are not only obvious ones ("I" conversion for integers, or "F" conversion for real numbers) but very special ones for very minute decimal places, mixtures of real numbers and exponents, etc. These last, however, are seldom needed by artists.

Normally the end of a program is indicated by the statement to STOP. If one were using one program as a subprogram of another, it might say RETURN (to the program from which the sub-program departed), but this means of ending a program is somewhat less common in applications to graphics, poetry, music. The last statement of all programs is END which is a convention telling the machine that no further calculations are required of it, and it may move on to

(continued on page 11)

E62808M TENNEY WORDS FORTRAN SOURCE LIST

ISN	SOURCE STATEMENT
0	$IBFTC WORDS LIST,REF
1	DIMENSION IPT(5),LINE(4)
2	DATA IPT/5H ,5H HANK,5H SHOT,5H MARY,5H DEAD/
3	WRITE(6,100)
4	J=0
5	DO 50 N1=1,5
6	LINE(1)=IPT(N1)
7	DO 50 N2=1,5
10	LINE(2)=IPT(N2)
11	DO 50 N3=1,5
12	LINE(3)=IPT(N3)
13	DO 50 N4=1,5
14	LINE(4)=IPT(N4)
15	J=J+1
16	IF(MOD(J,10)-1)40,45,40
17 40	WRITE(6,102)(LINE(N),N=1,4)
24	GO TO 20
25 45	WRITE(6,103)J,(LINE(N),N=1,4)
32 50	CONTINUE
37 100	FORMAT(1H1,20X,39H HANK AND MARY, A LOVE STORY, A CHORALE/43X,16H 1 BY DICK HIGGINS///)
40 102	FORMAT(10X,4A5)
41 103	FORMAT(16,4X,4A5)
42	STOP
43	END

FORTRAN IV program for Hank and Mary as realized by Dick Higgins and James Tenney.

ISNO. For labelling, the program was called simply WORDS. "List" asks the machine to print out the program, for convenient reference or to check for error. "Ref" causes a cross-reference index of each variable.

ISN1. This DIMENSION statement asks the machine to reserve five and only five possible locations in its "memory" for variable "IPT" and four for variable "LINE."

ISN2. The fact that there are only four words and a blank is fed into the machine. The "5H" means the machine is to read and store in "memory" the next five characters of each IPT, with the blank coming before each four-letter word in order to keep them from printing too close together. Note that the first word consists of five blanks.

ISN3. This WRITE statement simply causes the machine to head the printout with the full title. The number "6" referred to is, by convention, reserved for printed output (as opposed to tape, for example). The number "100" refers to the format statement numbered 100, which causes the title to be printed, in which 1H1 simply starts a new page. "20X" means 20 blanks will be printed before the title, causing an indentation. "39H" indicates the 39 letters (counting blanks as characters) of the full title, etc.

ISN4. "J=0" simply places variable J at the starting point [of which we shall see more later.] J itself is the index to the number of lines that have been printed.

ISN5. It is at this point that we enter a nest of DO loops. N1, N2, N3 and N4 are the indices of the four DO loops.

ISN6. This statement places the five possible values of the first member of the set LINE, designated LINE(1), successively into correspondence with the five members of set IPT. ISN10, 12 and 14 do the same in their respective DO loops.

ISN7. This statement sets up the structure parallel to ISN5 for the second unit in each line, as do ISN11 and ISN13 in theirs. Note that N2 has nothing to do with N1 — the names have been made similar in order to emphasize their parallel uses.

ISN15. Each time we fill in the four members of the line, we want to see how many lines we have done, whether or not we actually want to number all the lines. "J" — as mentioned in ISN4 — indicates the number of the line. Here each time we wish to increase the number of each line by one. If, hypothetically, we wanted to increase the number of each line by two, as written (not as executed), we would write "J=J+2."

ISN16. This stage decides if the line is to be numbered. It causes the machine to test for the value of J by using an IF statement. The statement IF(J)40,45,40 (hypothetically again) would exemplify the IF structure in its simplest form, which follows the convention of providing three possible courses, depending on whether J is negative (sending the machine to statement 40), zero (sending the machine to statement 45), or positive (sending it, again, to statement 40) respectively. The statement IF(MOD(J,10)-1)40,45,40 is a rather tricky concept, testing not just if a given value is to route the machine to statement 40, 45, or 40, but whether or not this given value, if diminished by one, is an even multiple of ten. If it is, it will cause the machine to refer to a WRITE statement that uses a FORMAT statement which includes numbering the line. If it isn't, the machine will proceed to a WRITE statement that refers to a FORMAT statement which doesn't include numbered lines.

ISN17. (statement 40 of the program). This instruction tells the machine to write out the lines per FORMAT statement 102. LINE has only four members, and these are to be written out in order, from member 1 to member 4. Note that there is no instruction to write out J.

ISN24. Having written out the line, we are ready to calculate the next line. But we must therefore by-pass the second WRITE statement which is reserved for the lines in which the value of J is listed. So we simply route the machine on to ISN32 (statement 50).

ISN25. As we noted in discussing ISN24, this is the WRITE statement which includes printing out J. However, it requires a slightly different FORMAT to refer to, because the machine must be told how and where to write the value of J.

ISN32. CONTINUE sends the machine back to the beginning of the DO loop nested closest to the end of the nest structure. If, of course, the five possible values in that loop have all been used up, it moves on to the next outer loop and starts the inside loop again; if both all the N4 and N3 values have been used up, it changes N2 and starts both N4 and N3 loops again, etc.

ISN37 (statement 100). We have already discussed this FORMAT statement in our discussion of ISN3. But I would like to point out the way the line carries over. The "1" listed causes the information to continue past the 72-character maximum which a single IBM card can accommodate.

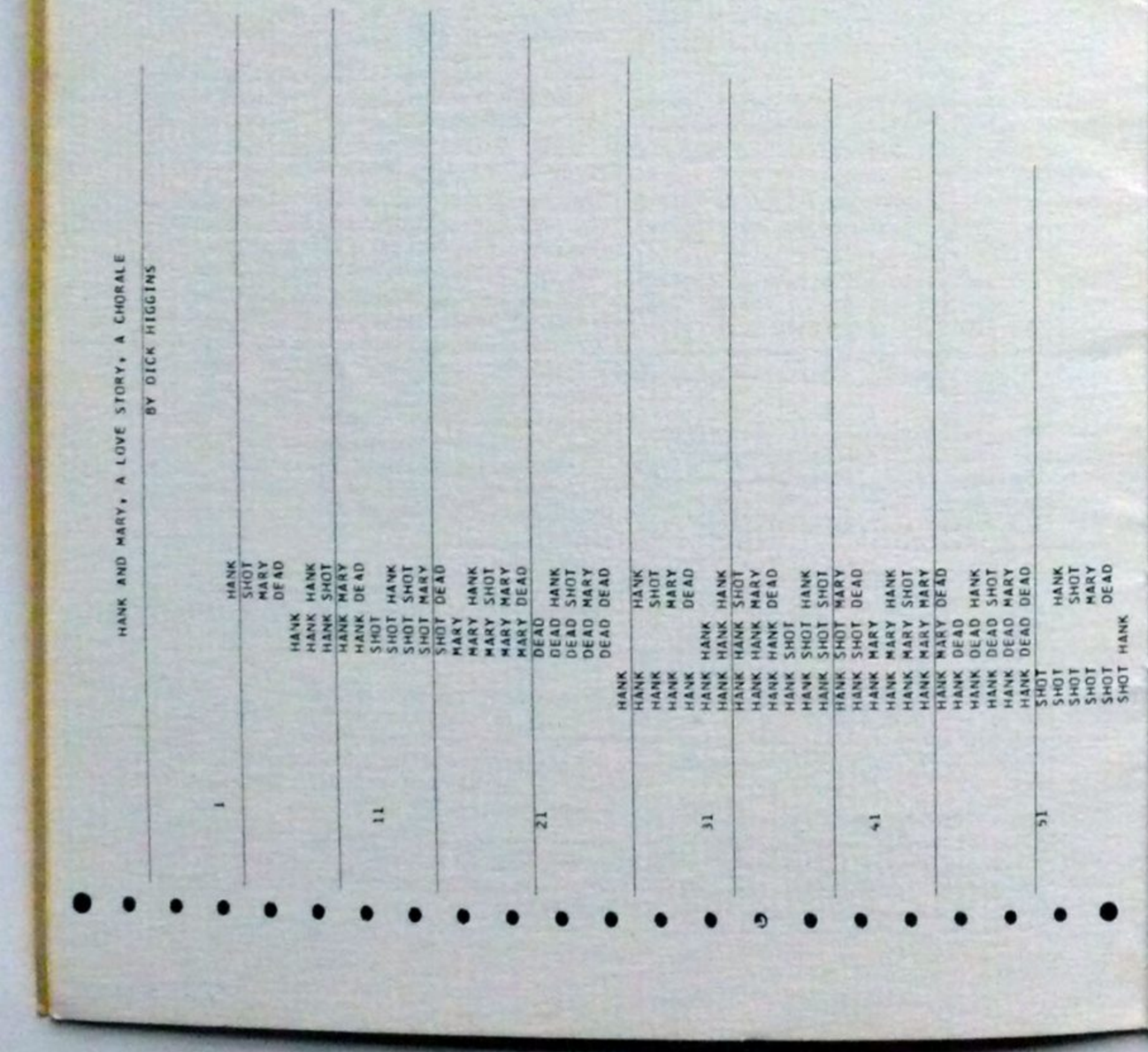
ISN40. The 10X of the format causes the machine to print ten spaces, so as to allow the word to align in columns. The 4A5 means that there will be four units of five characters each (see ISN2).

ISN41 (statement 103). Here we have told the machine to print J, which will always be an integer, with up to six numbers (more than needed, but this is how it is customarily done), followed by 4X (four blanks) totalling the 10 blanks we called for in ISN40.

ISN42 and 43. All FORTRAN IV programs end in this way, by convention, to indicate to the machine that there are no more instructions in this program.

Printout of hank and mary **as realized by Dick Higgins and James Tenney.**

Since computer time (the amount of time needed to rate a particular program) is extremely expensive, it is billed not by the job but by the minute. Note that the time at the end is shown to be 1.64 minutes, as opposed to the 16 hours needed to make the original typewritten version.



90c

SBN - 91185603X

Other ABYSS Publications include:

Towards The 1970's by Dick Higgins

Charles Bukowski: A Critical and Bibliographical Study by Hugh Fox

Countdown On An Empty Streetcar by Hugh Fox

Other Dick Higgins books include:

What Are Legends

Jefferson's Birthday/Postface

FOEW & OMBWHNW

A Book About Love & War & Death (Canto One)

A Book About Love & War & Death (Cantos Two & Three)

Towards The 1970's

Abyss Publications
Post Office Box C, Somerville, Mass. 02143