Chapter 7. FORTRAN and Some Other Languages

Vulgar languages that want
Words, and sweetness, and be scant
    Of true measure,
…
He that first invented thee,
May his joints tormented be,
    Cramp'd for ever;
Ben Jonson (1573 – 1637), *A Fit*
    of Rhyme Against Rhyme.

Introduction

In this chapter we shall introduce very briefly a half a dozen or so of the better known and historically important higher-level programming languages that have been developed to simplify the task of programming. We shall begin with FORTRAN which was the first widely used such language and finish with one or two of the currently more popular and fashionable languages, For each of these languages we shall give a few remarks about the early development of the language and a sample program written in the language. The sample program is for tabulating the results of throwing a die with an arbitrary number of faces an arbitrary number of times As sample data we shall consider the results of throwing a four-sided die 12 times giving the sequence 1, 4, 2, 3, 1, 1, 3, 3, 4, 2, 3, 4 of faces occurring which may be summarized in the following table:

1 3

2 2

3 4

4 3

The format of the data input to each program will be the list of faces preceded by an integer giving the number of faces on the die, and followed when necessary – and it is almost always necessary! - with a 0 indicating the end of data so that the sample data will be given as the following list:

4, 1, 4, 2, 3, 1, 1, 3, 3, 4, 2, 3, 4, 0

All of the sample programs are given in the Appendix to this chapter which begins with the sample program for the simple machine-language simulator of the last chapter.

In **J** the sample program may be written as

```
Freq=: (pos@[) ,. [: +/"1 pos@[ =/ ]
```

where `pos=: >:@i.` gives the positive integers as is illustrated by the following dialogue:

```
   pos 6
1 2 3 4 5 6
   4 Freq 1 4 2 3 1 1 3 3 4 2 3 4
1 3
2 2
3 4
4 3
```

First, though, we shall make a few remarks on the some of the very early languages which preceded FORTRAN.

Before FORTRAN

The first person to consider the design of a higher-level language appears to have been Konrad Zuse who was mentioned at the beginning of the last chapter as the designer and builder of the first programmable electromechanical computer. His work on the design of the language was begun as a graduate student but was not completed until after World War II. He considered that his language, which he called the Plankalkül, was an extension of the propositional claculus and predicate calculus of the German mathemtician David Hilbert. Zuse described his work in a lengthy manuscript which included programs for sorting, binary integer arithmetic, floating-point arithmetic, and forty-nine pages of programs for playing chess. Except for brief excerpts this work was not published until 1972 A brief summary was published in English in the same year and a complete English translation in 1976.

At about the time that Zuse was completing his work John von Neumann and colleagues developed a method of representing algorithms graphically using what became to be known as flow diagrams. This work was not formally published but appeared in their widely circulated *Planning and Coding of Problems for an Electronic Computing Instrument*. It had a significant influence on the development of the flow diagram as an important aid in the description of algorithms prior to their implementation in an executable programming language.

The first implemented programming language was Short Code which was developed for the UNIVAC, a computer intended for both commercial and scientific applications and designed and built for the Eckert-Mauchly Computer Corporation. The original purpose of Short Code was to have a program that would accept algebraic expressions in the form in which they were originally written, but unfortunately this goal was never attained.

The first implemented program that translated algebraic expressions into machine-language and which was used in programming real problems was Autocode written for the University of Manchester Mark I computer. The name Autocode was applied to several languages written for British computers in

the 1950s and the early 1960s. The first implemented compiler to receive extensive usage was the A-2 developed in 1955 for the UNIVAC. Possibly of greater importance was an interpretive algebraic system developed in the early 1950s at M.I.T. which was the first system in the United States to permit the user to write mathematical expressions in a form resembling conventional notation.

Now we turn to a very brief, and possibly idiosyncratic, look at a few of the more popular programming languages, almost all of which have been used in introductory courses, in the Department of Computing Science at the University of Alberta. Of course some of these languages, especially FORTRAN, have also been used throughout the University for research purposes. For some languages we shall give more than one version of the program in order to illustrate how the language has evolved. We note that all of the progrms have been debugged, sometimes with help duly acknowledged, except those for the University of Waterloo Watfiv version of FORTRAN and Algol W.

FORTRAN

In 1954 IBM formed a small group headed by John Backus to develop an automatic programming system for the IBM 704 that would produce efficient object code that would execute at a speed about the same as hand-generated code. The language was called FORTRAN for "Formula Translating (Language)". Emphasis was on efficiency of compilation and execution rather than on the design of the language. Indeed Backus later remarked that "we simply made up the langauge as we went along". The following excerpts from a report issued in November 1954 on the specifications of the language are of interest:

- The IBM Mathematical Formula Translating System or briefly, FORTRAN, will comprise a set of programs to enable the IBM 704 to accept a concise formulation of a problem in terms of a mathematical notation and to produce automatically a high-speed 704 program for the solution of the problem.
- Each future IBM calculator should have a system similar to FORTRAN accompanying it.
- It is felt that FORTRAN offers as convenient a language for stating problems for machine solution as is now available.
- After an hour course in FORTRAN notation the average programmer can fully understand the steps of a procedure stated in FORTRAN language without any additional comments.

FORTRAN I was available in 1956 for the IBM 704, and the first paper on the language appears to have been given at the Western Joint Computer Conference in 1957. In 1958 FORTRAN II was officially released for the 704 and shortly after for other IBM systems. Later versions of FORTRAN included FORTRAN IV released in 1964 and FORTRAN 77 and FORTRAN 90. The version of FORTRAN for the IBM 650, FORTRANSIT, may be of interest since it was an implemrntation of FORTRAN for a small

computer with a 2000-word memory. The complilation was a three-stage process which converted the original FORTRAN program first to IT (Internal Translator) which was a compiler developed at Purdue University, then to SOAP (Symbolic Optimal Assembly Program), and finally to machine language. Each stage produced a deck of punched cards which had to be removed from the card punch and used as input to the next stage. As compilation could be a lengthy process, some programmers would make corrections to the SOAP deck of cards rather than to the original source deck, thus omitting two stages of the compilation process.

The first sample FORTRAN program in the Appendix is written in the style of FORTRAN II and the second is for the University of Waterloo Watfiv.

BASIC

BASIC, for Beginner's All-purpose Symbolic Instruction Code, was developed at Dartmouth College, a small liberal arts college in Hanover, New Hampshire. (Incidentally, it was at a meeting of the American Mathematical Society at Dartmouth College in 1940 that George Stibitz of the Bell Telephone Laboratories first demonstrated the remote use of a computer over a communications line.) The BASIC language was developed as an alternative to FORTRAN for non-science students that would be "friendly", easy to learn and use, and convenient to access. It was designed by a group led by John G. Kemeny, who later became President of Dartmouth, and Thomas E. Kurtz. The first BASIC program under time-sharing was run at about 4:00 a.m. on May 1, 1964, and the first published program was the following:

```
10 LET X = (7+8)/3
20 PRINT X
30 END
```

BASIC proved to be an extremely popular language and now exists in many versions. Visual BASIC released in 1991 provided a simple means of constructing graphical user interfaces for using the language. BASIC may now be considered an almost universal language and has been made available on almost all computing systems. The two BASIC programs shown in the Appendix are in the style of the first BASIC and a more modern version and were both written using the QBASIC implementation of the language.

ALGOL

In 1958 a meeting of European and American computer specialists was held in Zurich to consider the design of an internationally recognized language. In the preliminary report of this meeting the language was referred to initially as IAL for "International Algorithmic Language" but this was soon changed to ALGOL for "Algorithmic Oriented Language" and the language described in the report as ALGOL 58. Once the preliminary report was published a number of mistakes and ambiguities were discovered and it

was realized that a completely new definition of the language would be required. Further discussions followed at a UNESCO-sponsered meeting in Paris in June 1959 and at another meeting, also in Paris, the following January. The final report defining the language, *Revised Report of the Algorithmic Language ALGOL 60*, was published in 1962 and 1963. In the report ALGOL 60 was described as "a language suitable for expressing a large class of numerical processes in a form sufficiently concise for direct automatic translation into the language of programmed automatic computers".

The syntax of the language was formally defined in the ALGOL 60 report using a notation developed by John Backus who headed the group within IBM that developed FORTRAN. The notation is referred to as BNF which is an acronym for either "Backus Normal Form" and for "Backus-Naur Form" because of the contribution of Peter Naur of the Regencentralen, Copenhagen to the ALGOL 60 report. BNF will be discussed briefly later in this chapter.

ALGOL W was similar in many ways to ALGOL 60 but contained many improvements. It was implemented at Stanford University on the IBM System/360. The ALGOL W program for the dice-throwing example has not been checked out since an ALGOL W compiler is no longer avaialable.

ALGOL 68, as the name implies, is in the ALGOL family of languages, and was designed and implemented over a number of years starting in the 1960s. It was a language which was designed by a committee whose membership was, initially at least, about two-thirds European, and which some people termed an unruly committee which finally broke up in disarray, We mention ALGOL 68 here only to pay tribute to a colleague, Barry Mailloux. Barry received a B.Sc. and an M.Sc. from the University of Alberta, and then went to the Mathematisch Centrum, Amsterdam where he wrote his doctoral dissertation on Algol 68.. He was one of the authors of the *Final draft report on the algorithmic language ALGOL 68* published in 1968. He then returned to the University of Alberta where he joined the Department of Computing Science, Sadly his career was cut short through illness and he died of a brain tumour in 1982.

Pascal

Pascal was originally developed by Niklaus Wirth of the Swiss Federal Institute of Technology (ETH) in Zurich as a language that would be efficient to implement and execute and that could be used for teaching the important concepts of computer programming. It was named after Blaise Pascal, the seventeenth-century French philosopher and mathematician. The language had its origins in Algol 60 but with additional features that allowed programmers to define their own data structures such as lists, trees and graphs. The language was also defined formally using Backus-Naur Notation. The following comments by Wirth on the language are worthy of note and have a generality far beyond the Pascal language:

> The desire for a new language for the purpose of of teaching programming is due to my deep dissatisfaction with the presently used languages whose features and constructs too often cannot be explained logically and convincingly and which too often represent an insult to minds trained in systematic reasoning. Along with this dissatisfaction goes my conviction that the language in which the student is taught to express his ideas profoundly influences his habits of thought and invention, and that the disorder governing these languages directly imposes itself on to the programming style of the students.

C, Java and Perl

The C programming language was developed at the Bell Telephone Laboratories in the early 1970s. It was derived from a language named B, which was derived from an earlier language BCPL. Originally C was designed as a systems language for the UNIX environment but was soon used for a large variety of applications. The C++ language, also developed at the Bell Telephone Laboratories, might be described as a superset of C with added features supporting classes.

Java was developed in the mid-1990s and owes much of its design to C and C++ although it has fewer low-level features than these languages. It was originally intended for writing programs for computer chips embedded in consumer electronic devices. However, it soon was found ideal for the design and implementation of programs intended for distribution and use on the Web. Many users consider Java to be one of the most important computer languages developed so far.

Perl, an acronym for “Practical Extraction and Report Language”, was developed in the mid-1980s as a text-processing language for the easy and efficient manipulation of various types of text files. It is now also used for graphics programming, system administration and network programming. As with Java many of its features are similar to those of C.

MATLAB

MATLAB was developed in the late 1970s at the University of New Mexico and Stanford University and was originally intended for use in courses in matrix algebra, linear algebra and numerical analysis. It has been described as a high-level technical computing language implemented in an interactive environment for algorithm development, data visualization and analysis and numerical computation. There are add-on “toolboxes”, i.e., application packages, for extending the use of MATLAB to such areas as signal and image processing, financial modelling and computational biology.

Spreadsheets

The first spreadsheet, VisiCalc (for Visible Calculator), was developed as a tool for financial analysis.

It was the work of Daniel Bricklin, a Harvard MBA student, and a programmer friend, Bob Frankson. It was an immediate success when it was released in 1979 as an application for the Apple II computer. Since then some form of spreadsheet software has been made available for every personal computer and has become the indispensable tool of almost all computer users. The sample program has been implemented using the Works spreadsheet, a simplified version of the widely used Microsoft Excel spreadsheet, to illustrate that the spreadsheet may be considered a programming language which often presents a simpler solution to a problem than does a conventional programming language.

Backus-Naur Form

Backus-Naur Form, also known as Backus Normal Form, with the acronym BNF which stands for either name, was introduced earlier in the chapter as a means of formally describing the syntax of ALGOL 60 and other programming languages. We note that BNF describes only the *syntax* of a lnguage; the *semantics* or the meaning attached to the symbols must be described separately. In this section we shall describe BNF briefly and give an example of its use in the definition of a constant.

The syntactic units or variables being described are enclosed in pointed brackets < and > to distinguish them from the symbols used in the language itself. The symbol ::= is used to denote equivalence of the terms appearing on its right and left, a vertical bar | is used to indicate “or” while “and” is denoted by the juxtaposition of symbols. We can define a constant as follows:

<digit> ::= 0|1|2|3|4|5|6|7|8|9

<unsigned integer> ::= <digit>|<unsigned integer><digit>

<integer> ::= <unsigned integer>|+<unsigned integer>|-<unsigned integer>

<exponent> ::= E<integer>

<decimal> ::= <integer>.<unsigned integer>|<integer><exponent>|

<integer>.<unsigned integer><exponent>

< constant> ::= < integer>|<decimal>

The first statement defines the ten digits 0, 1, 2, …, 9, and the second defines an unsigned integer as any sequence of these digits. We note that the second definition is recursive, as are many definitions in BNF, since the item being defined occurs on both sides of the defining equation. These definitions define, for example, 123, 3.14159, 1E6 and 0.314159E1as constants but not .314159 or E6.

Acknowledgements

I would like to thank the following persons for their assistance with some of the sample programs: Maria Stepanova for debugging the FORTRAN II program and for reminding me that Watfiv was not compatible with current versions of FORTRAN, Roman Fedoriw for debugging the C and Perl programs,

and Laura Watson for debugging my first (and I hope last) Java program. It was a pleasure working with these people and their assistance was much appreciated

Appendix – Example programs for dice frequencies

***** Machine language *****

```
   00 859   Read N = No. of faces
   01 459   )
   02 259   ) Counter = 0
   03 558   )
   04 451   }
   05 158   } Set clear instruction
   06 509   }
   07 459   )
   08 259   ) Clear freq.
   09 5--   )
   10 458   }
   11 150   } Incr. Counter
   12 558   }
   13 259   )
   14 704   ) Is Counter < N?
   15 860   Read d = dice face
   16 460   )
   17 260   ) -d
   18 260   )
   19 721   Is -d < 0?
   20 631   Transfer to printing
   21 452   )
   22 160   ) Set pick-up instr.
   23 527   )
   24 453   }
   25 160   } Set put-away instr.
   26 529   }
   27 4--   )
   28 150   ) Incr. freq.
   29 5--   )
   30 615   Repeat
   31 450   ) Counter = 1
   32 558   )
   33 454   }
   34 158   ) Set print instr.
   35 537   )
   36 958   Print Counter
   37 9--   Print freq.
   38 458   )
   39 259   ) Is Counter < N?
   40 742   )
   41 000   Halt
   42 458   )
   43 150   ) Incr. Counter
   44 558   )
   45 633   Repeat
   46   0
   47   0
   48   0
   49   0
   50   1   )
   51 561   )
   52 460   ) Program constants
   53 560   )
   54 960   )
```

C FORTRAN II

```
      DIMENSION IFREQ(20)
  100 FORMAT(5X,2I5)
      READ(5,100) N
      DO 1 I=1,N
      IFREQ(I)=0
    1 CONTINUE
    4 READ(5,100) NUM
      IF(NUM) 2,2,3
    3 IFREQ(NUM)=IFREQ(NUM)+1
      GO TO 4
    2 DO 5 I=1,N
      WRITE(6,100) I,IFREQ(I)
    5 CONTINUE
      STOP
      END
```

```
C     WATFIV FORTRAN
      INTEGER FREQ(20),D
  100 FORMAT(5X,2I5)
      READ(5,100) N
      DO 1 I=1,N
      FREQ(I)=0
    1 CONTINUE
      READ(5,100) D
      WHILE(D .GT. 0) DO
         FREQ(D)=FREQ(D)+1
         READ(5,100) D
      END WHILE
      DO 2 I=1,N
      WRITE(6,100) I,FREQ(I)
    2 CONTINUE
      STOP
      END
```

```
100 REM FIRST BASIC PROGRAM
110 DIM FREQ(20)
120 DATA 4,1,4,2,3,1,1,3,3,4,2,3,4,0
130 READ N
140 FOR I = 1 TO N
150    FREQ(I) = 0
160 NEXT I
170 READ D
180 IF D = 0 THEN 210
190 FREQ(D) = FREQ(D) + 1
200 GOTO 170
210 FOR I = 1 TO N
220    PRINT I, FREQ(I)
230 NEXT I
240 STOP
250 END
```

```
REM SECOND BASIC PROGRAM
DIM FREQ(20)
DATA 4,1,4,2,3,1,1,3,3,4,2,3,4,0
READ N
FOR I = 1 TO N
   FREQ(I) = 0
NEXT I
READ D
WHILE D > 0
   FREQ(D) = FREQ(D) + 1
   READ D
WEND
FOR I = 1 TO N
   PRINT I, FREQ(I)
NEXT I
STOP
END
```

```
COMMENT
   ALGOL W;
BEGIN
INTEGER ARRAY FREQ(1::20);
INTEGER D,N;
READ(N);
FOR I:=1 UNTIL N DO FREQ(I):=0;
READON(D);
WHILE D > 0 DO
   BEGIN
   FREQ(D):=FREQ(D) + 1;
   READON(D);
   END;
FOR I:=1 UNTIL N DO WRITE(I,FREQ(I));
END.
```

```
(* Pascal *)
program DiceFrequencies(input,output);
type a = array[1..20] of integer;
var i,d,n  : integer;
    freq   : a;
begin
read(n);
for i:= 1 to n do freq[i]:= 0;
read(d);
while d > 0 do
   begin
   freq[d]:= freq[d] + 1;
   read(d);
   end;
for i:= 1 to n do writeln(i,'  ',freq[i]);
end.
```

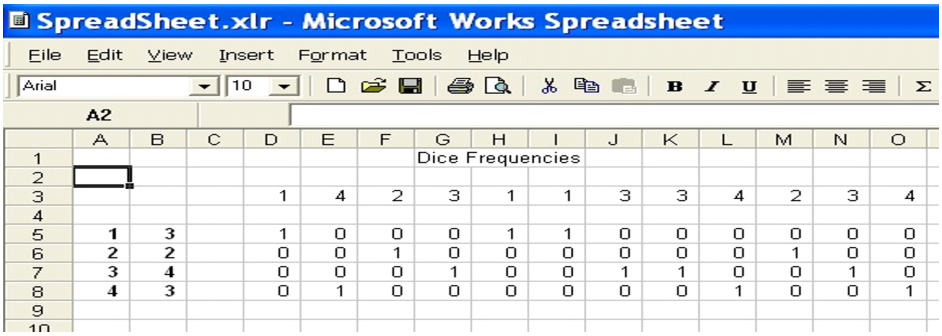
```
main() /* C */
  {
     int d,i,n;
     int freq[21];
     scanf("%d",&n);
     for (i = 1; i < n; ++i)
        freq[i] = 0;
     scanf("%d",&d);
     while (d > 0) {
        ++freq[d];
        scanf("%d",&d);
     }
     for (i = 1; i <= n; ++i)
        printf("  %d   %d\n",i,freq[i]);
   }
```

```
// Java
public class DiceFrequencies {
   public static void main (String arg[]) {
   int d, i;
   int freq[] = new int [21];
   int D[] = {4,1,4,2,3,1,1,3,3,4,2,3,4,0};
   int N = D[0];
   for (i = 1; i <= N; ++i)
      freq[i] = 0;
   i = 1;
   while (D[i] > 0) {
      d = D[i];
      freq[d] = ++freq[d];
      i = ++i;
      }
   for (i = 1; i <= N; ++i)
      System.out.println (i + " " + freq[i]);
   }
}
```

```
# Perl
@D = (4,1,4,2,3,1,1,3,3,4,2,3,4,0);
$N = shift@D ;
foreach (1..$N){
   $freq[$_] = 0;
}
$d=shift@D ;
while ($d > 0) {
   $freq[$d] +=1;
   $d=shift@D ;
}
foreach (1..$N){
   printf("%5d %5d \n", $_ , $freq[$_] );
```

```
% MATLAB
N = 4;
D = [1 4 2 3 1 1 3 3 4 2 3 4];
Face = [1:N]';
Freq = zeros(N,1);
for i = 1:N
   Freq(i) = sum(D == i);
end
disp([Face Freq])
```

***** Works spreadsheet *****



SpreadSheet.xlr - Microsoft Works Spreadsheet

File Edit View Insert Format Tools Help

Arial 10

A2

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1							Dice Frequencies								
2															
3				1	4	2	3	1	1	3	3	4	2	3	4
4															
5	**1**	**3**		1	0	0	0	1	1	0	0	0	0	0	0
6	**2**	**2**		0	0	1	0	0	0	0	0	0	1	0	0
7	**3**	**4**		0	0	0	1	0	0	1	1	0	0	1	0
8	**4**	**3**		0	1	0	0	0	0	0	0	1	0	0	1
9															
10															