12-1-1982

Computer generation of Pascal program: an experiment in artificial intelligence

Tze-Ping Angela Yang
Atlanta University

Follow this and additional works at: http://digitalcommons.auctr.edu/dissertations
Part of the Mathematics Commons

Recommended Citation
COMPUTER GENERATION OF PASCAL PROGRAM:
AN EXPERIMENT IN ARTIFICIAL INTELLIGENCE

by

TZE-PING ANGELA YANG

A thesis submitted in practical fulfillment of the requirements for the degree of
Master of Science
(Computer Science)

at
Atlanta University
Mathematics Department
December, 1982
CONTENTS

Chapter 1. Introduction ........................................ 1
  Artificial intelligence (AI) ............................... 1
  Objectives of AI .......................................... 4
  Some applications in AI .................................. 6

Chapter 2. Program background ............................... 9
  Pascal language ........................................... 9
  The process of program development .................... 10

Chapter 3. Program description ............................. 13
  Programming techniques .................................. 13
  Purpose of my program .................................. 14

Chapter 4. Program documentation ......................... 20
  Program environment .................................... 20
  Program run preliminaries ............................... 20
  Compilation statistics .................................... 21
  Memory requirements ..................................... 21
  User's qualification ...................................... 21
  Program procedures ..................................... 21
  1. Menu .................................................. 21
  2. Procedure source ..................................... 26
  3. Procedure dataprocessing ............................. 33
  4. Procedure destination ................................. 37
Recommendation ......................... 42
Bibliography ................................. 43
Appendix A. The LAST.PAS program ............... 45
Appendix B. Running processing and output of the LAST.PAS program ............... 63
Chapter 1
Introduction

Artificial Intelligence (AI)

Can a machine think? It is an interesting question. Research scientists in Artificial Intelligence (AI) have a dream to build thinking machine just the same as the invention of the computer itself. They try to get machines to exhibit behavior that we call intelligence behavior when we observe it in human beings. Many human activities, such as solving problems, playing games, understanding and translating languages, doing mathematics, proving theorems, and even driving an automobile, are said to demand "intelligence."

Almost all the machines to be discussed are computers. By a computer is meant a high-speed general-purpose store-program electronic computer. If computers could perform tasks such as intelligence, then presumably, this computer would possess some degree of "artificial intelligence." It is hard to get a precise and formal definition of AI. But, briefly speaking:

Artificial intelligence is the part of computer science concerned with designing intelligence computer systems, that is,
systems that exhibit the characteristics we associate with intelligence in human behavior--understanding languages, solving problems, and so on.1

It may be useful to subdivide the whole area of AI into two branches. One of these may be simplistically called the "engineering approach"; the other, the "modeling approach." In the first case, the researcher wants to create a system that is able to deal with interesting and difficult intellectual tasks, regardless of whether the methods and techniques used are similar or identical to those used by humans. He has a job to accomplish inexpensively, efficiently, and in a reliable manners. Some applications of this approach are the translation of text from one natural language into another, composition of music by the computer, selection of sites of warehouses across the country in an optimum manner, and so on.

The modeling approach has the basic research objective of trying to gain an understanding of the mechanisms of a real life system and to explain and predict its behavior. We can put in this category, for example,

those projects that simulate human problem solving, decision making, or learning behavior (illustrated by models of neural networks).

But, how can a computer do these tasks? When we talk about people who do these intelligent things, we usually talk about which they have to know in order to do them. On the other hand, we describe someone's ability to behave with intelligence in terms of his or her knowledge. For this reason, a computer has to own some abilities in order to do the intelligent work. The computer programs, which are designed by programmers can assign some knowledge to the machine. We say that a computer program can be designed to know how to play cards, or understand spoken English, or manipulate a robots. We say that programs know about objects in its domain, about events that have taken place, or about how to perform specific tasks.

As a result, AI systems use knowledge about a problem to suggest shortcut solution--a technique known as heuristic problem--solving. AI researchers developed methods for efficiently representing and processing facts and idea on a computer.
Objections of AI

The central goal of work in artificial intelligence is to make computers intelligence, both to make them more useful and to understand the principles that make intelligence possible. That intelligent computers will be extremely useful is obvious. The more profound point is that artificial intelligence aims to understand intelligence using the methods of computation, thus offering a radically new and different basis for theory formation. The benefits of this new knowledge, most of the people who doing AI believe, will apply to any intelligent information processor, whether biological or solid state.

Generally speaking, we can briefly list the three basic motivations for research in artificial intelligence.

(1). To replace human intelligence because the latter or fragile, expensive, scarce, and often less than reliable.

(2). To establish theories of human intelligence in the form of simulation models.

(3). To assess the capabilities of presently available software and hardware, and to point to the development for future programming languages and computer systems.
Again, a fundamental objective is to write highly efficient programs of universal applicability. A plausible avenue to this goal is via learning programs that can initially tackle simple problems and gradually acquire more and more power for a large number of, and more difficult problems.

Dozens of computer programs have been invented whose performance begins to match that of humans. But, more recently, the field has emerged from speculation to the hard task of writing programs to perform various tasks commonly conceded to require intelligence if done by a human. The body of activity goes variously under the name of "artificial intelligence," "machine intelligence," sometimes more modestly, "complex information processing," as well as numerous other terms more descriptive of the particular type of application.

To get a general feeling for the way that computers have been programmed to perform tasks intelligently, we move to discussed some work in simulation of AI. It is clearly not possible to give a detail summary of the current work. However, I do give you a more general idea in AI activities.
Some applications in AI

AI researchers want to build a system that can make computers smart. They also expect to open vast new opportunities for automation in the office, factory, stores, and even home. Language plays an important role in our human life. We have to communicate with each other through language by speaking, and reading every day. The fact of our languages are what separates us from the beasts. For this reason, if we want machines that communicate with us as they outperform us, we need to teach them how to understand our languages. To accomplish this, AI researchers try to develop a "natural" language which is an artificial and highly simplified language. A natural language has its own linguistic and grammar. With greater ease, we can use it to communicate with computers.

Commercial systems that are able to understand ordinary English are beginning to come out. An English language information system, owned by Artificial Intelligence Corp. in Waltham, Mass. allows users to store information on a specific subject in a computer data base and later retrieve the information via questions typed in English on a keyboard.
For example, to retrieve information about engineers from an employee file, a user may type:

"Show me all the engineers in New York who earn less than $50,000."

The system will use linguistic knowledge in the form of a grammar to translate such question into formal queries. It may be:

"DISPLAY FILE = employee FOR JOB-DESCRIPTION = engineer AND SEX = male. AND CITY = New York AND SALARY < 50000"

The system's formal query language processor then retrieves and displays the answer on a CRT screen. A system like that is built by combining a natural language with a data-base management system. It can perform remarkably well because it takes advantage of the DBMS' ability to answer a wild range of questions about a subject. This is an example of computer that can understand human language and then translate it into the system languages.

In the effort to make machines think, some insights have been gained into how humans think, and therefore learn. AI researchers suggest some ways for educating schoolchildren to make it more effective and pleasant for everyone concerned--child, parent, and teacher.
For example, a computer-aided instruction (CAI) system engages children in a video game that requires the use of basic arithmetic skills, thereby sweetening the pill of learning. Another program diagnosis arithmetic mistakes by having a child do a series of subtraction problems designed to reveal faulty calculation procedures.

In addition to understanding languages and developing a teaching tool, AI can also be applied to many fields such as chemistry, biology, and so on. I have discussed some general activities that have already been developed. Perhaps one of the big payoff from AI is likely the valuable insight into the operation of human brain may result from study of the research on artificial intelligence. The Pascal program that I will deal with later is my application of AI.
Chapter 2
Program Background

We already have a general idea of meaning and significance of artificial intelligence. And we also know that computer programs can make machines do acts which require intelligence. The purpose of this thesis is to introduce the reader to a Pascal program, I call it "the last Pascal program" which can code another Pascal programs for the user.

Pascal language

Pascal is a popular language and use widely in school and university. It was developed by Wirth (1971) in the early 1970's following some earlier work by Wirth and Hoare (1960).

The development of the language Pascal is based on two principal aims. The first is to make available a language suitable to teach programming as a systematic discipline based on certain fundamental concepts clearly and naturally reflected by the language. The second is to develop implementations of this language which are both reliable and efficient on presently available computers.2

---

At this stage of evolution, Pascal is one of the powerful and elegant language, complex enough to introduce important concepts in computer programming but simply enough to be a good teaching tool in a course in computer programming. It is relatively easy for an unsophisticated student to learn and understand the Pascal language. For this reason, Pascal is popularly used at universities.

A student in a cost-free computer usage situation like AUC, is usually unaware of the cost-driving elements involved in the use of the computers. He only knows that he writes the program, types it into the computer, runs it, and then corrects the errors as many times as necessary until he gets the correct program results. It is important to let the student understand how needless error-correction wastes computer time; therefore it would be fruitful if we discuss some of issue involved in program development.

The process of program development

The Pascal language is a high-level language, which can not be directly understood by any computers. That is, no known computer can execute the commands of high-level languages directly. People write in these languages
because they are convenient. Their English-like words, their natural algebraic notation, and their use of the familiar decimal numbering system are helpful to the programmers. However, a program in one of the high-level languages, called a source program, must be translated into an equivalent program in the internal machine language of the computer. That program is called the object program. This process is called compilation, which is done by a computer program called a compiler. The input to the compiler program is a source program and the output of the compiler is the object program and a program listing. The program listing is a printed representation of the program that was presented to the compiler for translation. This listing may also include error messages, if any, and other information from the compiler about the translation process.

After compilation we begin the second phase of running a program on the computer: execution. During this step, the computer will sequentially execute the machine language commands contained in the object program. Any input data required by the object program will now be read and any output produced will be printed.

Once a programmer gets the result, he has to debug
the program to investigate whether the result is correct or not. The process of debugging involves discovering, locating, and correcting all errors that cause a program to produce either incorrect results or sometimes no result at all. After debugging, of course, he has to repeat the preceding of two phases: compilation and execution. These same tasks will do over and over again until he gets the desired result.

The compilation and execution are inevitable. The way that improves the process of program development is to avoid the debugging phase. That is my purpose of doing "the last Pascal program." The next chapter will discuss it in detail.
Chapter 3
Program Description

Programming techniques

A sample program:

```pascal
PROGRAM ADD(INPUT,OUTPUT);
VAR
    A,B,C:INTEGER;
BEGIN
    READLN(A,B);
    C:=A+B;
    Writeln(C:8)
END.
```

When you take the course of Pascal language in a university, the sample program above may be the first one that your teacher assigns to you. This program is very simple: compute the sum of any two integer numbers. Before you get this assignment, you may have already known the format of a Pascal program look like. And your teacher must tell you the process of doing a program. It might have the following steps:

1. Define the problem you will solve and then write an algorithm or draw a flowchart.

2. Code the program.

3. Type the program into the computer and run it.
(4) Test and debug the program.

Purpose of my program

The person who is learning the Pascal language or who already knows it must have an impression that when he does his first program, he will spend more of his time repairing the program than he does writing it, although it is short and simple. He spends a significant amount of his time reworking a faulty program. He wonders why it is necessary to code a program to solve such a simple problem. He can easily get the result of the sum of any two integer numbers by hand.

Because of being unfamiliar with the Pascal language, a beginner may have two kinds of errors: syntax error and logical error. Syntax errors are violations of grammatical rules of the language. They are easier to correct than most other kind, because they usually produce an error message which indicates what is wrong with that statement. These error messages are produced by the Pascal compiler which translates a source program into an object program. Any grammatical mistakes will cause problems when the translation is performed. Logical errors usually cause the incorrect
result or sometimes no result at all. Programs with complex, intricate, and "jump" logic such as a loop, conditional or unconditional jumps are much more difficult to work with and to debug. Because these statements do not execute sequentially according to the program listing. They modify the flow of control by skipping a group of statements or by returning to the statements that have already executed. Programmers must follow the flow of control through the program to look for the errors.

Generally, the beginning programmer does not realize that program debugging will be the single most time-consuming phase in the overall program development process. Studies on the development of large programming systems indicate that it is not uncommon for 50% to 75% of the overall programming time to be spent of finding and correcting errors. This percentage is probably accurate for student jobs as well.

We know that a computer is invented to handle work at a rate faster than that of humans and with greater accuracy than humans. It saves time and simplifies routine work. And we say that computer programs have the ability to make machines do something "intelligence."

Can a computer write the program like the sample above
automatically for people. Of course, it can. In fact, the subject of this thesis is such a program. The output of this program will be a Pascal program which is free of all errors. We do not worry about missing semicolon at the end of statement, misspelling the reserved words, or not declaring the variable, and so on. You get the impression that the computer knows all of the grammar and syntax of Pascal. It is hoped that we do not need coding, typing, debugging, and testing procedures any more. Because the computer can do them, with the aid of my program, for us. That is the reason why I called it "the last Pascal program." Because after mine, nobody is required to code Pascal program any more. If you need Pascal program, just let "the last Pascal program" help you.

Of course, the use of "the last Pascal program" does not eliminate the need for sound planning in program development. In fact, a chief motivation of my thesis is to relieve the programmer of the coding and debugging burden in favor of more time to devote to the program design activities. So, when we write a program, we must find out exactly what it is supposed to do. In other words, we should consider as primary the design of
a solution to our problem. This is problem definition. The problem definition phase involves developing and clarifying the exact specifications of the program. Some programmers frequently begin to develop a program with an ill-defined and ill-conceived statement of the problem. This can lead to confusion, uncertainty and, worst of all, an incorrect solution.

Therefore, it is necessary to write a set of problem specifications that spell out in clear, unambiguous language that exact problem we are attempting to solve. These problem specifications will contain three important class of information. In my problem development, the programmer must consider design specification in each of the following areas.

(1) Input specifications. This part describes the input to the program. This may include answers to the following questions:

   a. What specific types will be represented by the input data?
   b. What format will the values be in?
   c. How will we know when we have exhausted the input? Will we be notified by a specific symbol or will we have to determine that for ourselves?
(2) Data processing specifications. When we write a program, we do so because the computer can do something for us. But, we should tell it what we wish to do. For example, we can tell it to add two numbers or calculate the average of ten different numbers. It is advisable to draw a flowchart or write an algorithm. A flowchart is a drawing showing the steps required in the solution of a problem, especially when a problem is complex. It is perhaps the best method available for expressing what computers can do—or what we will do. It can not only provide a detail blueprint to be used in writing a computer program but provide a way of communicating the program logic to other people.

(3) Output specifications. Just as we needed to specify the input provide to the program, we must describe in detail the output we will be provide. It may include answers to the following types of questions.

a. What values will be produced?

b. What is the format of these values?

c. What specific annotation, headers, or titles are required in the output?

Figure 1 is the flowchart of the last Pascal program. The next chapter will describe the program.
documentation so that the reader can understand it more detail and know how to use it.

Figure 1: flowchart of the last Pascal program.
Chapter 4
Program Documentation

Program environment

This program was designed to operate on the following hardware/software environment:

(1) Hardware: The program is designed to run on the VAX 11/780 system, manufactured by the Digital Equipment Corporation.

(2) Software: This program was implemented under the VMS (Version 2.4) operating system and was written in VAX Pascal language.

Program run preliminaries

The last Pascal program is in the file named "LAST.PAS." To run it, type "RUN LAST" following the "$" sign.

(See appendix B for running process.)

$ RUN LAST

The output from such a run is a Pascal program in accordance with your specifications. This desired program is in the file named "F2.PAS." To compile and run it, type the following commands after the "$" sign.

$ PAS (or PAS/NOS) F2
$ LINK F2
$ RUN F2
Compilation statistics

Total space allocated: 24566 bytes
Stack frame size total: 2409 bytes
Run time: 27.95 seconds (2187 line/minute)

Memory requirements

67 blocks for execution LAST program.
71 blocks for objection LAST program.
56 blocks for LAST Pascal program.
Total requirements are 194 blocks.

User's qualification

The person who knows at least one computer language (BASIC, or FORTRAN), should be able to use this program.

Program procedures

1. Menu

This program is divided into three phases. A menu shows these three steps in which the program will proceed. Each phase option only be used once during a run.
1.1

MENU
=====
(1) INPUT DATA
(2) DATA PROCESSING
(3) OUTPUT DATA
(4) QUIT

PLEASE CHOOSE A MENU OPTION BY THE NUMBERS.

1

THE SOURCE OF INPUT:
(1) FILE
(2) TERMINAL
(3) IN PROGRAM

PLEASE CHOOSE ONE OF NUMBERS.

.
.
.

If you choose the first menu--input data, you will get in the procedure source. See section 2 for more information.
MENU

(1) INPUT DATA
(2) DATA PROCESSING
(3) OUTPUT DATA
(4) QUIT

PLEASE CHOOSE A MENU OPTION BY THE NUMBERS.

2

COMPUTE
(1) IN SAME RECORD.
(2) BETWEEN RECORDS.

PLEASE CHOOSE ONE OF NUMBERS.

.
.
.

If you choose the second menu--data processing, you will get in the procedure dataprocessing. See section 3 for more information.
1.3

MENU

(1) INPUT DATA
(2) DATA PROCESSING
(3) OUTPUT DATA
(4) QUIT

PLEASE CHOOSE A MENU OPTION BY THE NUMBERS.

3

THE DESTINATION OF OUTPUT:
(1) FILE
(2) TERMINAL

PLEASE CHOOSE ONE OF NUMBERS.

If you choose the third menu--output data, you will get in the procedure destination. See section 4 for more information.
1.4

MENU
======

(1) INPUT DATA
(2) DATA PROCESSING
(3) OUTPUT DATA
(4) QUIT

PLEASE CHOOSE A MENU OPTION BY THE NUMBERS.

4

PLEASE ENTER THE DESIRED PROGRAM NAME (<=30 CHAR) ?

.
.
.

Once you choose the last menu--quit, it means you finish three processes and you want to stop it. The program name, which is for program heading will be asking you. You can choose any name you like using letters from A to Z or digits from 0 to 9. The first character should be letter. Do not exceed 30 characters for the whole name.
2. Procedure source

This procedure asks users two questions:

(1) the source of input
(2) input data map

2.1 The source of input

There are three possible input sources:

(1) from a file
(2) from terminal
(3) in program
Figure 2: flowchart of procedure source.
2.1.1 File

THE SOURCE OF INPUT:
(1) FILE
(2) TERMINAL
(3) IN PROGRAM

PLEASE CHOOSE ONE OF NUMBERS.
1

FILE NAME:
IN.DAT

INPUT DATA MAP:

Suppose you choose the first source--file(sourceno =1), it means your input will come from a file. You have to tell the computer your file name so that it can code an OPEN and RESET procedures for your program. In the example above, IN.DAT is the file name that you tell the computer. See section 2.1.3 for input data map.
2.1.2 Terminal

THE SOURCE OF INPUT

(1) FILE:
(2) TERMINAL
(3) IN PROGRAM

PLEASE CHOOSE ONE OF NUMBERS.

2

INPUT DATA MAP

...

If you choose the second source (sourceno=2), it indicates your input is from a terminal. See next section for input data map.

2.1.3 Input data map

The input data map is the description of input data. It specifies the type and format of input data. If your source is from a file or terminal, you have to describe this information.
INPUT DATA MAP:

THIS DATA MAP IS THE DESCRIPTION OF INPUT DATA. IT INCLUDES TWO PARTS:
(1) DATA TYPE: THERE ARE THREE DATA TYPES: INTEGER, REAL AND CHARACTER. TYPE I, R, OR C TO REPRESENT THEM.
(2) COLUMN LOCATION: INPUT DATA FORMAT.

TYPE "Q", IF FINISH.

```
<table>
<thead>
<tr>
<th>DATA TYPE</th>
<th>COLUMN LOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>1-2</td>
</tr>
<tr>
<td>I</td>
<td>4-5</td>
</tr>
<tr>
<td>R</td>
<td>7-9</td>
</tr>
<tr>
<td>Q</td>
<td></td>
</tr>
</tbody>
</table>
```

YOU HAVE FOLLOWING VARIABLES:
I1, I2, R1

HOW MANY RECORDS DO YOU HAVE?
5

The purpose of this map is to let the computer know:
(1) the number of variables your program have according to the data type so that it can declare these variables and their types.
(2) the order of these variables according to the column location so that it can code the READ statement.
A data map describes one line (record) of data. You may have more than one lines (records) but have same format; therefore you should tell the computer the exact number of records.

Like the example above, it indicates you have five records in input. This can let the computer know how many times the input statement will execute.
In program

THE SOURCE OF INPUT:
(1) FILE
(2) TERMINAL
(3) IN PROGRAM

PLEASE CHOOSE ONE OF NUMBERS.

3

PLEASE ENTER THE DATA.

NOTICE:
(1) EACH LINE CAN NOT EXCEED 80 CHARACTERS.
(2) IF MORE THAN ONE VALUE, USE A SPACE TO SEPARATE THEM.
(3) IF CHARACTER, USE A QUOTATION MARK TO COVER IT.
(4) TYPE "Q", IF FINISH.

12 34
I1:=12;
I2:=34;
3.4 8.0
R1:=3.4;
R2:=8.0;
'ABC'
L1:='ABC';
Q

The third possible input source is in program (sourceno=3). In this case, we can say you do not have input and just assign some values to variables. You just have to tell the computer your desired values like the example above.

Once the computer reads in these data, it will check their types (integer, real, or char) so that it can declare the variables, and then assign these values
to appropriate variables. You have to keep in mind these variable names because you will use them later in data processing phase.

3. Procedure dataprocessing

This procedure asks users what kind of processing they want to do. For example, computation, sort, or condition check, and so on. My program can be available users to do the computation. Figure 3 is the flowchart of this procedure.

There are two different cases:

(1) do the computation in same record.

or (2) do it between records.

Suppose you have following data:

<table>
<thead>
<tr>
<th>real 1</th>
<th>integer 1</th>
<th>integer 2</th>
<th>real 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>col 1-3</td>
<td>5-6</td>
<td>8-9</td>
<td>11-13</td>
</tr>
<tr>
<td>2.0</td>
<td>29</td>
<td>10</td>
<td>1.2</td>
</tr>
<tr>
<td>3.7</td>
<td>23</td>
<td>51</td>
<td>0.3</td>
</tr>
<tr>
<td>1.9</td>
<td>25</td>
<td>39</td>
<td>3.1</td>
</tr>
</tbody>
</table>

...
Figure 3: flowchart of procedure dataprocessing.
3.1 Compute in same record

    COMPUTE
    (1) IN SAME RECORD.
    (2) BETWEEN RECORDS.

PLEASE CHOOSE ONE OF NUMBERS.

1

DO YOU NEED LOOP IN PROGRAM? (Y OR N)
Y
J IS LOOP INDEX.

WITHIN LOOP FORMULA(S):
NOTICE:
(1) EACH FORMULA CAN NOT EXCEED 50 CHARACTERS.
(2) ONE LINE FOR ONE FORMULA.
(3) TYPE "Q", IF FINISH.

I1+I2
I3 WILL BE THE RESULT OF THIS FORMULA.
R1+R2
R3 WILL BE THE RESULT OF THIS FORMULA.
I3*R3
R4 WILL BE THE RESULT OF THIS FORMULA.
Q

This proceed indicate you only use the same line
data to consist of the formula. Before you enter the
formula, you should have converted the algebraic formula
into the Pascal form. In the example above, you can also
combine the set three formulas with only one: (I1+I2)*(R1+R2),
and the result will be in R3.
3.2 Compute between records

```
COMPUTE
(1) IN SAME RECORD.
(2) BETWEEN RECORDS.

PLEASE CHOOSE ONE OF NUMBERS.

2

(1) ADD ONE VARIABLE ALL RECORDS.
(2) AVERAGE OF ONE VARIABLE ALL RECORDS.

PLEASE CHOOSE ONE OF NUMBERS.

1

VARIABLE(S):
(1) ONE LINE FOR ONE VARIABLE.
(2) TYPE "Q", IF FINISH.
I1
R1
Q
YOU HAVE FOLLOWING VARIABLES:
SUM1 SUM2
```

Suppose you have hundreds of records, and you want to compute the sum or average of them. That is the meaning you compute between records. In this case, the computer asks you to compute sum or average, and then asks the variable names you want to compute.

In the example above, it will add 29, 23, 25... to SUM1 and add 2.0, 3.7, 1.9... to SUM2. Same case for doing average; only change result variables to AVE1 and AVE2.
4. Procedure destination

This procedure asks users two questions:

(1) the destination of output
(2) output data map

Figure 4 is the flowchart of this procedure.

4.1 The destination of output

There are two possible destinations:

(1) in a file
(2) in terminal
Figure 4: flowchart of procedure destination.
4.1.1 File

THE DESTINATION OF OUTPUT:
(1) FILE
(2) TERMINAL

PLEASE CHOOSE ONE OF NUMBERS.

1

FILE NAME:
OUT.DAT

OUTPUT DATA MAP:

.
.
.

If you choose the first one--file(outno=1), it means your output data will be in a file. The same as input, you should tell the computer your file name so that it can code an OPEN and REWRITE procedures in program. See section 4.1.3 for output data map.
4.1.2 Terminal

THE DESTINATION OF OUTPUT
(1) FILE
(2) TERMINAL

PLEASE CHOOSE ONE OF NUMBERS.

2

OUTPUT DATA MAP:

If you choose the second one—terminal(outno=2), it means that the terminal will show you the output data when you run the program. See next section for output data map.

4.1.3 Output data map

No matter which destination you choose, you have to tell the computer your output format. The output data map allows you to show what information you want to print out.
OUTPUT DATA MAP:

------------------------------------------------------------------------
THIS DATA MAP IS THE DESCRIPTION OF OUTPUT FORMAT.
IT INCLUDES TWO PARTS:
(1) VARIABLE OR STRING: IF VARIABLE, TYPE I1, R1, OR C1 TO REPRESENT
THE FIRST INTEGER, REAL, OR CHAR.
IF STRING, USE A QUOTATION MARK TO COVER IT.
(2) SPACE: SPACE NUMBERS FOR THIS VARIABLE OR STRING.
IF REAL, "TYPE BOTH WHOLE SPACE NUMBERS AND THE
SPACE NUMBER AFTER DECIMAL POINT: USE A COLON TO
SEPARATE THEM.
------------------------------------------------------------------------

TYPE "Q", IF FINISH.

OUTPUT DATA MAP

<table>
<thead>
<tr>
<th>VARIABLE OR STRING</th>
<th>SPACE</th>
</tr>
</thead>
<tbody>
<tr>
<td>I3</td>
<td>6</td>
</tr>
<tr>
<td>R3</td>
<td>8:1</td>
</tr>
<tr>
<td>R4</td>
<td>8:1</td>
</tr>
<tr>
<td>Q</td>
<td></td>
</tr>
</tbody>
</table>

In the example above, it specifies to print out
three values: I3, R3, and R4. These three variables
come from the formulas:

I3 = I1 + I2;
R3 = R1 + R2;
R4 = I3 + R3;
Recommendation

My program is only a first step, because it can only generate the computation programs and each program can have only one loop and input/output statements. But, when programs are complex, they may require more than one loops or input/output statements. It is necessary to include some more functions in order to make the program more flexible and powerful.

The following four functions are important, because they will be useful when a program is complex. It is my suggestion that append these functions to the "last Pascal program."

(1) TYPE declaration.

(2) Subprograms (procedures and functions).

(3) IF-THEN-ELSE statement.

(4) More than one loops and input/output statements.
BIBLIOGRAPHY


Winston, Patrick Henry and Brown, Richard Henry ed.  
Artificial Intelligence: An MIT Perspective.  
Appendix A

The LAST.PAS program
PASCAL PROGRAMS FOR PEOPLE.

AUTHOR: TZE-PING A, YANG

DATE: OCT., 1982

BEGIN

OPENFILE(OUTFILE, 'F1, DAT', HISTORY := NEW);
REWRITE(OUTFILE);
OPEN(OUTFILE, 'F2, PAS', HISTORY := NEW);
REWRITE(OUTFILE);

PROCEDURE PRINT3O(STRING: NAME);
VAR
J: INTEGER;
BEGIN
J := 1;
WHILE STRING[J] <> '' DO
BEGIN
WRITE(OUTFILE, STRING[J]);
J := J + 1
END;
IDENTIFY (* PRINT3O *)

PROCEDURE PRINT4O(STR2: VARGRTYP);
VAR
M: INTEGER;
BEGIN
FOR M := 1 TO 40 DO
BEGIN
IF (STR2[M] <> '') THEN
WRITE(OUTFILE, STR2[M])
ELSE IF (M + 1 <= 40) THEN
PROCEDURE PRINT60(STR:READEXP); R L:INTEGER; GIN FOR L:=1 TO 60 DO BEGIN IF STR[L] <> ' ' THEN WRITE(OUTF,STR[L]) ELSE IF (L+1 <= 60) THEN IF STR[L+1] <> ' ' THEN WRITE(OUTF,STR[L]) END D; (* PRINT60 *)

PROCEDURE PRINT601(S1:READEXP); R GIN FOR L:=1 TO 60 DO BEGIN IF S1[L] <> ' ' THEN WRITE(OUTFILE,S1[L]) ELSE IF (L+1 <= 60) THEN IF S1[L+1] <> ' ' THEN WRITE(OUTFILE,S1[L]) END D; (* PRINT60 *)

PROCEDURE PRINTBLANK(N:INTEGER); R GIN FOR E:=1 TO N DO BEGIN WRITE(OUTFILE,' ') END D; (* PRINTBLANK *)

**********INPUT DATA MAP**********

PROCEDURE DATAMAP; THIS PROCEDURE IS TO BUILD AN INPUT DATA MAP, WHICH IS THE DESCRIPTION OF INPUT DATA, IT INCLUDES TWO PARTS: DATA TYPE (INTEGER, REAL, OR CHAR) AND COLUMN LOCATION.

DATATYPE,DASH:CHAR; LOW,HIGH:INTEGER; GIN
WRITELN(' INPUT DATA MAP:'); WRITELN('-----------------------------------------------'); WRITELN(' THIS DATA MAP IS THE DESCRIPTION OF INPUT DATA. '); WRITELN(' IT INCLUDES TWO PARTS: '); WRITELN(' (1) DATA TYPE: THERE ARE THREE DATA TYPES: INTEGER, REAL');
WRITELN("INPUT DATA MAP");
WRITELN("----------------------");
WRITELN("TYPE "Q", IF FINISH.");
WRITELN("----------------------");
WRITELN("DATA TYPE
COLUMN LOCATION");
WRITELN("----------------------");
READ(DATATYPE);
WHILE DATATYPE <> "Q" DO
BEGIN
  DASH := CHR(13);
  READ(LOW, DASH);
  IF DASH = "" THEN
    READ(HIGH);
  READLN;
  CASE DATATYPE OF
    "I": BEGIN
      I := I + 1;
      IF I < 10 THEN
        WRITE(OUTF, "I", I:1)
      ELSE WRITE(OUTF, "I", I:2)
    END;
    "R": BEGIN
      R := R + 1;
      IF R < 10 THEN
        WRITE(OUTF, "R", R:1)
      ELSE WRITE(OUTF, "R", R:2)
    END;
    "C": BEGIN
      IF DASH = "" THEN
        BEGIN
          CC := CC + 1;
          IF CC < 10 THEN
            WRITE(OUTF, "L", CC:1)
          ELSE WRITE(OUTF, "L", CC:2)
          NO[CC] := HIGH - LOW + 1
        END; "CASE #*
    END;
  END;
  READ(DATATYPE);
  IF DATATYPE <> "Q" THEN
    WRITE(OUTF, ");
END; (* WHILE LOOP *)
WRITELN(OUTF);
CLOSE(OUTF);
OPEN(OUTF, "F1.DAT", HISTORY := OLD);
RESET(OUTF);
READLN(OUTF, READSTRING); WRITELN;
WRITELN("YOU HAVE FOLLOWING VARIABLES");
WRITELN(READSTRING); WRITELN;
OPEN(OUTF,"I1.DAT",HISTORY:=OLD);
REWRITE(OUTF);
ND; (* DATAMAP *)

*********OUTDATAMAP**********
PROCEDURE OUTDATAMAP;
&
UNVAR:ARRAY[1,60] OF CHAR;
G:INTEGER;
COLON:CHAR
SCN
FOR G:=1 TO 15 DO
BEGIN
SPACENO[G]:=0;
SPACENO1[G]:=0
END;
WRITELN;
WRITELN(" OUTPUT DATA MAP:" );
WRITELN;
WRITELN("---------------------------------------------------");
WRITELN(" THIS DATA MAP IS THE DESCRIPTION OF OUTPUT FORMAT, ");
WRITELN(" IT INCLUDES TWO PARTS:" );
WRITELN(" (1) VARIABLE OR STRING: IF VARIABLE, TYPE I, R, OR C ");
WRITELN(" TO REPRESENT THE FIRST INTEGER, REAL, OR CHAR. ");
WRITELN(" (2) SPACE: SPACE NUMBERS FOR THIS VARIABLE OR STRING, ");
WRITELN(" IF REAL, TYPE BOTH WHOLE SPACE NUMBERS AND THE ");
WRITELN(" SPACE NUMBER AFTER DECIMAL POINT; USE A COLON TO ");
WRITELN(" SEPARATE THEM. ");
WRITELN;
WRITELN(" TYPE"Q", IF FINISH." ); WRITELN;
WRITELN(" OUTPUT DATA MAP:" );
WRITELN;
WRITELN(" VARIABLE OR STRING SPACE" );
WRITELN("---------------------------------------------------");
GO:=1;
READ(VARORSTREGGO); UNPACK(VARORSTREGGO, UNVAR,1);
IF (ORD(UNVAR[1])<39) AND (UNVAR[1]<>"J") AND (UNVAR[1]<>"I")
BEGIN
WRITELN("WRONG VARIABLE! TRY AGAIN.");
READLN(VARORSTREGGO); UNPACK(VARORSTREGQQ, UNVAR,1)
END;
WHILE UNVAR[1] <> "Q" DO
BEGIN
IF (UNVAR[1] = "L") OR (UNVAR[1] = "C") THEN
BEGIN
READLN;
FLAG2[GO]:=3
END
ELSE BEGIN
READ(SPACENO[GO]);
IF ORD(UNVAR[1])=39 THEN
ELSE CASE UNVAR[1] OF
  'I', 'J': FLAG2[GO] := 0;
  'R', 'A', 'S': FLAG2[GO] := 1
END;
IF FLAG2[GO] = 1 THEN READLN(COLON, SPACE1, GO)
ELSE READLN
END;
WRITELN('----------------------------------------');
GO := GO + 1;
READ(VARORSTR[GO]);
UNPACK(VARORSTR[GO], UNVAR, 1)
END; (* WHILE O LOOP *)
GO := GO - 1
END; (* OUTDATAMAP *)

(***************DECLAREVARIABLE**********************)
PROCEDURE DECLAREVARIABLE;
(* THIS PROCEDURE IS TO DEFINE VARIABLES AND THEIR DATA TYPE ACCORDING TO THE DATA MAP. *)
VAR
  X: INTEGER;
BEGIN
  IF (I <> 0) OR (R <> 0) OR (C <> 0) OR (CC <> 0) THEN BEGIN
    WRITELN(OUTFILE, 'VAR');
    IF I <> 0 THEN BEGIN
      WRITE(OUTFILE, ' I');
      IF I >= 10 THEN BEGIN
        FOR X := 2 TO 9 DO WRITE(OUTFILE, ', I', X:1);
        FOR X := 10 TO I DO WRITE(OUTFILE, ', I', X:2)
      END ELSE BEGIN
        FOR X := 2 TO I DO WRITE(OUTFILE, ', I', X:1)
      END;
      WRITELN(OUTFILE, ': INTEGER;')
    END;
    IF R <> 0 THEN BEGIN
      WRITE(OUTFILE, ' R1');
      IF R >= 10 THEN BEGIN
        FOR X := 2 TO 9 DO WRITE(OUTFILE, ', R', X:1);
        FOR X := 10 TO R DO WRITE(OUTFILE, ', R', X:2)
      END ELSE BEGIN
        FOR X := 2 TO R DO WRITE(OUTFILE, ', R', X:1)
      END;
      WRITELN(OUTFILE, ': REAL;')
    END;
    IF C <> 0 THEN BEGIN
      WRITE(OUTFILE, ' C');
      IF C >= 10 THEN BEGIN
        FOR X := 2 TO 9 DO WRITE(OUTFILE, ', C', X:1);
        FOR X := 10 TO C DO WRITE(OUTFILE, ', C', X:2)
      END ELSE BEGIN
        FOR X := 2 TO C DO WRITE(OUTFILE, ', C', X:1)
      END;
      WRITELN(OUTFILE, ': CHARACTER;')
    END;
    IF SPACEN <> 0 THEN BEGIN
      WRITE(OUTFILE, ' SPACEN');
      IF SPACEN >= 10 THEN BEGIN
        FOR X := 2 TO 9 DO WRITE(OUTFILE, ', SPACEN', X:1);
        FOR X := 10 TO SPACEN DO WRITE(OUTFILE, ', SPACEN', X:2)
      END ELSE BEGIN
        FOR X := 2 TO SPACEN DO WRITE(OUTFILE, ', SPACEN', X:1)
      END;
      WRITELN(OUTFILE, ': INTEGER;')
    END;
    IF COLON <> 0 THEN BEGIN
      WRITE(OUTFILE, ' COLON');
      IF COLON >= 10 THEN BEGIN
        FOR X := 2 TO 9 DO WRITE(OUTFILE, ', COLON', X:1);
        FOR X := 10 TO COLON DO WRITE(OUTFILE, ', COLON', X:2)
      END ELSE BEGIN
        FOR X := 2 TO COLON DO WRITE(OUTFILE, ', COLON', X:1)
      END;
      WRITELN(OUTFILE, ': INTEGER;')
    END;
    IF SPACE <> 0 THEN BEGIN
      WRITE(OUTFILE, ' SPACE');
      IF SPACE >= 10 THEN BEGIN
        FOR X := 2 TO 9 DO WRITE(OUTFILE, ', SPACE', X:1);
        FOR X := 10 TO SPACE DO WRITE(OUTFILE, ', SPACE', X:2)
      END ELSE BEGIN
        FOR X := 2 TO SPACE DO WRITE(OUTFILE, ', SPACE', X:1)
      END;
      WRITELN(OUTFILE, ': INTEGER;')
    END;
    IF 0 <> 0 THEN BEGIN
      WRITE(OUTFILE, ' 0');
      IF 0 >= 10 THEN BEGIN
        FOR X := 2 TO 9 DO WRITE(OUTFILE, ', 0', X:1);
        FOR X := 10 TO 0 DO WRITE(OUTFILE, ', 0', X:2)
      END ELSE BEGIN
        FOR X := 2 TO 0 DO WRITE(OUTFILE, ', 0', X:1)
      END;
      WRITELN(OUTFILE, ': INTEGER;')
    END;
  END;
END;
IF C >= 10 THEN
BEGIN
FOR X := 2 TO 9 DO
WRITE(OUTFILE,'"C",X:1);
FOR X := 10 TO C DO
WRITE(OUTFILE,'"C",X:2)
END;
ELSE BEGIN
FOR X := 2 TO C DO
WRITECOUTFILE,"C",X:1)
END;
IF CC <> 0 THEN
BEGIN
FOR X := 1 TO CC DO
BEGIN
IF X >= 10 THEN
WRITELNCOUTFILE,~L~,X:2,"PACKED ARRAY[1,"NO[X]:2,"] OF CHAR;
END
END
END
ND; (* DECLAREVARIABLE *)

PROCEDURE OPENIN;
BEGIN
WRITECOUTFILE,"OPEN(INFILE,");
WRITE(OUTFILE,CH);
PRINT30(FILENAME1);
WRITECOUTFILE,CH;
WRITELN(OUTFILE,"HISTORY:QLD");
WRITELN(OUTFILE,"RESET(INFILE);")
ND; (* OPENIN *)

PROCEDURE OPENOUT;
BEGIN
WRITECOUTFILE,"OPEN(OUTFILE,");
WRITE(OUTFILE,CH);
PRINT30(FILENAME2);
WRITECOUTFILE,CH;
WRITELN(OUTFILE,"HISTORY:NEW");
WRITELN(OUTFILE,"REWRITE(OUTFILE);")
ND; (* OPENOUT *)

***************DESTINATION**************
PROCEDURE DESTINATION;
* THIS PROCEDURE IS TO ASK USER WHERE TO PUT THE
OUTPUT DATA,
THERE ARE TWO POSSIBLE DESTINATIONS:
(1) FILE
(2) TERMINAL  *

PROCEDURE FILEOUT;
BEGIN
WRITELN("FILE NAME: ");
READLN(FILENAME2);
OUTDATAMAP
ND; (* FILEOUT *)
**BEGIN OUTDATAMAP;**
**WRITELN**
**END;** (* OUTTERMINAL *)

**BEGIN (** OUTDESTINATION **)**
**WRITELN**
**WRITELN**("THE DESTINATION OF OUTPUT:");
**WRITELN**(" (1) FILE ");
**WRITELN**(" (2) TERMINAL");
**WRITELN**;
**WRITELN**("PLEASE CHOOSE ONE OF NUMBERS.");
**WRITELN**;
**READLN**(OUTNO);
**IF** (OUTNO <> 1) **AND** (OUTNO <> 2) **THEN**
**BEGIN**
**WRITELN**("WRONG NUMBER! TRY AGAIN.");
**READLN**(OUTNO)
**END;**
**CASE** OUTNO **OF**
1:*FILEOUT;
2:*OUTTERMINAL
**END**
**END;** (* DESTINATION *)

(***************SOURCE***************

**PROCEDURE** SOURCE;
(* THIS PROCEDURE IS TO ASK USER WHERE TO GET THE INPUT DATA, THERE ARE THREE POSSIBLE SOURCES: (1) FILE (2) FROM TERMINAL (3) IN PROGRAM *)

**PROCEDURE** FILEIN;
**BEGIN**
**WRITELN**("FILE NAME:");
**READLN**(FILENAME);
**DATAMAP**
**END;** (* FILEIN *)

**PROCEDURE** FROMTERMINAL;
**BEGIN**
**DATAMAP**;
**WRITELN**
**END;**

**PROCEDURE** INPROGRAM;
**LABEL** 30;
**VAR**
DATAIN:PACKED ARRAY[1..80] OF CHAR;
T,P,G,K:INTEGER;
CHECKTYPE:BOOLEAN;
UNREADEXP:ARRAY[1..60] OF CHAR;
UNDATAIN:ARRAY[1..80] OF CHAR;

**PROCEDURE** CHECKNUM(N:INTEGER);
**VAR**
X:INTEGER;
**BEGIN**
BEGIN
  WRITE(N:2,"=''");
  WRITE(OUTF,N:2,"=''");
END
ELSE BEGIN
  WRITE(N:1,"=''");
  WRITE(OUTF,N:1,"=''");
END;
FOR X=T TO P-1 DO
BEGIN
  WRITE(DATAIN[X]);
  WRITE(OUTF,DATAIN[X])
END;
WRITELN();
WRITELN(OUTF,"'');
ID; (* CHECKNUM *)
BEGIN
  P:=1;
  READLN(DATAIN);
  UNPACK(DATAIN,UNDATAIN,1);
  WHILE UNDATAIN[P] <> "Q" DO
BEGIN
  T:=P;
  REPEAT
    IF UNDATAIN[P] <> " " THEN
    BEGIN
      IF ORD(UNDATAIN[P]) <> ASP THEN
      BEGIN
        WHILE UNDATAIN[P] <> " " DO
        BEGIN
          P:=P + 1;
        END;
        FOR Q=T TO P-1 DO
        BEGIN
          IF DATAIN[Q]=',' THEN
          BEGIN
            CHECKTYPE:=TRUE;
            Q:=P
          END;
          ELSE CHECKTYPE:=FALSE;
        END; (* FOR Q LOOP *)
      END;
      IF CHECKTYPE THEN
      BEGIN
        WRITE("R");
        WRITE(OUTF,"R");
        R:=R + 1;
        CHECKNUM(R)
      END
      ELSE BEGIN
        WRITE("I");
        WRITE(OUTF,"I");
        I:=I + 1;
      END;
    END;
  END;
END;
WRITELN("PLEASE ENTER THE DATA.");
WRITELN("NOTICE:");
WRITELN("(1) EACH LINE CAN NOT EXCEED 80 CHARACTERS.");
WRITELN("(2) IF MORE THAN ONE VALUE, USE A SPACE TO SEPARATE THEM.");
WRITELN("(3) IF CHARACTER, USE A QUOTATION MARK TO COVER IT.");
WRITELN("(4) TYPE "Q", IF FINISH.");
WRITELN;
WHILE DATAIN(P) <> ' ' DO
BEGIN
  P:=P + 1;
END;
IF (P-T-2)=1 THEN
BEGIN
  WRITE('C');
  WRITE(OUTF,'C');
  C:=C + 1;
  CHECKNUM(C)
END
ELSE BEGIN
  WRITE('L');
  WRITE(OUTF,'L');
  CC:=CC + 1;
  NO[CC]:=P - T - 2;
  CHECKNUM(CC)
END
P:=P + 1;
T:=P
END
ELSE GOTO 30;
UNTIL (P=80);
30:FOR Q:=1 TO 80 DO
  DATAIN[Q]=" ";
P:=1;
READLN(DATAIN);
UNPACK(DATAIN,UNDATAIN,1)
END;(* WHILE Q LOOP *)
WRITELN; WRITELN;
(* INPROGRAM *)

BEGIN
  WRITELN; WRITELN('THE SOURCE OF INPUT:');
  WRITELN(' (1) FILE ');
  WRITELN(' (2) TERMINAL ');
  WRITELN(' (3) IN PROGRAM ');
  WRITELN;
  WRITELN('PLEASE CHOOSE ONE OF NUMBERS.');
  WRITELN;
  READLN(SOURCENO);
  IF (SOURCENO <> 1) AND (SOURCENO <> 2) AND (SOURCENO <> 3) THEN
    BEGIN
      WRITELN('WRONG NUMBER! TRY AGAIN. ');
      READLN(SOURCENO)
    END;
  CASE SOURCENO OF
    1:FILFIN;
    2:FROMTERMINAL;
    3:INPROGRAM
  END
END; (* SOURCE *)

***************DATAPROCESSING***************
PROCEDURE DATAPROCESSING;

54
PROCEDURE SAMERECORD;
LABEL 40;
VAR
   UNPACKFOR:ARRAY[1..60] OF CHAR;
   G:INTEGER;
BEGIN (* SAMERECORD *)
   WRITELN('DO YOU NEED LOOP IN PROGRAM? (Y OR N)');
   READLN(YN);
   IF YN='Y' THEN
      BEGIN
         WRITELN('J IS LOOP INDEX.');
         IF SOURCENO=0 THEN
            BEGIN
               WRITELN('WHAT IS MAXIMUM VALUE OF J ?');
               READLN(RECORDNO)
            END;
         END;
      WRITELN('WITHIN LOOP FORMULA(S):')
   ELSE BEGIN
      WRITELN;
      END;
   WRITELN('NOTICE:');
   WRITELN('(1) EACH FORMULA CAN NOT EXCEED 50 CHARACTERS,');
   WRITELN(' (2) ONE LINE FOR ONE FORMULA,' );
   WRITELN(' (3) TYPE "O", IF FINISH.');
   READLN(FORMULA);
   UNPACK(FORMULA,UNPACKFOR,1);
   WHILE UNPACKFOR[1] <> 'G' DO
      BEGIN
         G:=1;
         WHILE UNPACKFOR[G] <> ' ' DO
            BEGIN
               IF UNPACKFOR[G] = 'R' THEN
                  BEGIN
                     FLAG:=1;
                     GOTO 40;
                  END;
               ELSE FLAG:=0;
               GI:=G + 1;
            END;
      40:IF FLAG=1 THEN
         BEGIN
            R:=R + 1;
            IF R >= 10 THEN
               BEGIN
                  WRITELN('R',R:2,' IS THE VARIABLE OF THIS FORMULA,' );
                  WRITE(OUTF,'R',R:2,':="')
               END;
            ELSE BEGIN
                  WRITELN('R',R:1,' IS THE VARIABLE OF THIS FORMULA,' );
                  WRITE(OUTF,'R',R:1,':="')
               END;
            PRINT60(FORMULA);
            WRITELN(OUTF,';')
         END;
      ELSE BEGIN
         I:=I + 1;
         IF I >= 10 THEN

BEGIN
  WRITELN('"I",1:2," IS THE VARIABLE OF THIS FORMULA.');
  WRITE(OUTF,'"I",I:2)
END
ELSE BEGIN
  WRITE(OUTF,':=');
  PRINTS0(FORMULA);
  WRITELN(OUTF,:"")
END;
READ(OUTF);
UNPACK(FORMULA,UNPACKFOR,1)
END
END; (* SAME RECORD *)

PROCEDURE AMONGRECORD;
VAR
  K:INTEGER;
  UNNAME:ARRAY[1..3] OF CHAR;
BEGIN
  WRITELN;
  WRITELN(' (1) ADD ONE VARIABLE ALL RECORDS.');
  WRITELN(' (2) AVERAGE OF ONE VARIABLE ALL RECORDS.');
  WRITELN('PLEASE CHOOSE ONE OF NUMBERS.');
  WRITELN;
  READLN(BENO);
  IF (BENO <> 1) AND (BENO <> 2) THEN
  BEGIN
    WRITELN('WRONG NUMBER! TRY AGAIN.');
    READLN(BENO)
  END;
  WRITELN;
  WRITELN('VARIABLE(S):');
  WRITELN(' (1) ONE LINE FOR ONE VARIABLE.');
  WRITELN(' (2) TYPE "Q", IF FINISH.');
  WRITELN;
  GV:=0;
  REPEAT
    GV:=GV + 1;
    READLN(VARNNAME[GV],UNNAME,1);
    UNPACK(VARNNAME[GV],UNNAME,1);
    BEGIN
      WRITELN('YOU TYPE WRONG VARIABLE! TRY AGAIN.');
      READLN(VARNNAME[GV]);
      UNPACK(VARNNAME[GV],UNNAME,1)
    END;
    UNTIL UNNAME[1]="Q";
  GV:=GV - 1;
  WRITELN('YOU HAVE FOLLOWING VARIABLES FOR THE RESULT:');
  FOR K:=1 TO GV DO
  BEGIN
    IF BENO=1 THEN
    BEGIN
      IF GV >= 10 THEN
        WRITE(' SUM',K:2)
      ELSE WRITE(' SUM',K:1)
    END
    ELSE BEGIN
      ...
    END
IF GW >= 10 THEN
  WRITE(' AVE',K:2)
ELSE WRITE(' AVE',K:1)
END;
WRITELN;
END; (* AMONGRECORD *)

BEGIN (* DATAPROCESSING *)
  IF (SOURCENO=3) OR (SOURCENO=0) THEN
    DATANO:=1
  ELSE BEGIN
    WRITELN;
    WRITELN('COMPUTE');
    WRITELN(' 1 IN SAME RECORD.');
    WRITELN(' 2 BETWEEN RECORDS.');
    WRITELN('PLEASE CHOOSE ONE OF NUMBERS.');
    WRITELN;
    READLN(DATANO);
    IF (DATANO <> 1) AND (DATANO <> 2) THEN
      BEGIN
        WRITELN('WRONG NUMBER! TRY AGAIN,');
        READLN(DATANO)
      END;
    END;
    CASE DATANO OF
      1:SAMERECORD;
      2:AMONGRECORD
    END
END; (* DATAPROCESSING *)

PROCEDURE CODEWRITE;
VAR
  O:INTEGER;
BEGIN
  PRINTBLANK(BLANKNO);
  WRITE(OUTFILE,' ',OUTFILE,'');
  IF OUTNO=1 THEN
    WRITE(OUTFILE,' ',OUTFILE,'');
  O:=0;
  O:=O + 1;
  IF FLAG2(O)=1 THEN
    BEGIN
      PRINT4O(VARORSTR[O]);
      WRITE(OUTFILE,' ',OUTFILE,'');
    END
  ELSE BEGIN
    PRINT4O(VARORSTR[O]);
    IF FLAG2(O)=0 THEN
      WRITE(OUTFILE,' ',OUTFILE,'');
    END;
  IF O < GO THEN
    WRITE(OUTFILE,' ',OUTFILE,'');
  UNTIL O >= GO;
  WRITELN(OUTFILE,' ',OUTFILE,'');
END; (* CODEWRITE *)

PROCEDURE CODESTATE12;
VAR
G,K,M:INTEGER;
PROCEDURE CODESTART;
BEGIN
  IF GV <> 0 THEN
    BEGIN
      FOR M:=1 TO GV DO
      BEGIN
        PRINTBLANK(BLANKNO);
        IF M >= 10 THEN
          WRITELN(OUTFILE,"SUM",M:2,"=0;"
        ELSE WRITELN(OUTFILE,"SUM",M:1,"=0;"
      END;
      PRINTBLANK(BLANKNO);
      WRITE(OUTFILE,"WHILE NOT EOF*;
      IF SOURCENO=1 THEN
        WRITELN(OUTFILE,"INFILE")
      ELSE WRITELN(OUTFILE,"DO")
      PRINTBLANK(BLANKNO);
      WRITELN(OUTFILE,"BEGIN")
      BLANKNO:=BLANKNO + 3;
      PRINTBLANK(BLANKNO);
      WRITE(OUTFILE,"READLN")
      IF SOURCENO=1 THEN
        WRITE(OUTFILE,"INFILE")
      PRINT30(READSTRING);
      WRITELN(OUTFILE,";")
      FOR M:=1 TO GV DO
      BEGIN
        PRINTBLANK(BLANKNO);
        IF M >= 10 THEN
        WRITELN(OUTFILE,"+',VARNAME[M],";"
      END;
      END;
    END;
  END; (* CODESTART *)

PROCEDURE READFILESTATE;
BEGIN
  IF SOURCENO <> 0 THEN
    BEGIN
      PRINTBLANK(BLANKNO);
      WRITE(OUTFILE,"READLN(";
      IF SOURCENO=1 THEN
        WRITE(OUTFILE,"INFILE")
      PRINT30(READSTRING);
      WRITELN(OUTFILE,";")
    END;
  WHILE NOT EOF(OUTF) DO
  BEGIN
    PRINTBLANK(BLANKNO);
    READLN(OUTF,READEXP);
    PRINT601(READEXP);
    WRITELN(OUTF)
  END;
END;
(* READFILESTATE *)

BEGIN
  IF SOURCENO=2 THEN
  BEGIN
    END;

58
PRINTBLANK(BLANKNO);
WRITELN(OUTFILE,"PLEASE ENTER THE DATA,",CH,"");
END;
IF DATANO = 0 THEN
CODEWRITE
ELSE BEGIN
IF DATANO=1 THEN
BEGIN
IF (RECORDNO > 1) AND (YN='Y') THEN
BEGIN
PRINTBLANK(BLANKNO);
WRITELN(OUTFILE,"FOR J:=1 TO ",RECORDNO:2,' DO");
PRINTBLANK(BLANKNO);
WRITELN(OUTFILE,"BEGIN");
BLANKNO:=BLANKNO + 3;
READFILESTATE;
BLANKNO:=BLANKNO - 3;
PRINTBLANK(BLANKNO);
WRITELN(OUTFILE,"END;")
END
ELSE READFILESTATE
END
ELSE BEGIN
IF SOURCENO = 2 THEN
BEGIN
PRINTBLANK(BLANKNO);
WRITELN(OUTFILE,"TYPE "Z,IF FINISH,"",CH,",");
END;
IF BENO <> 0 THEN
IF BENO=1 THEN
BEGIN
CODESTART;
BLANKNO:=BLANKNO - 3;
PRINTBLANK(BLANKNO);
WRITELN(OUTFILE,"END;"');
CODEWRITE
END
ELSE BEGIN
PRINTBLANK(BLANKNO);
WRITELN(OUTFILE,"ACC:=0;");
CODESTART;
PRINTBLANK(BLANKNO);
WRITELN(OUTFILE,"ACC:=ACC + 1;");
BLANKNO:=BLANKNO - 3;
PRINTBLANK(BLANKNO);
WRITELN(OUTFILE,"END;"');
FOR M:=1 TO GV DO
BEGIN
PRINTBLANK(BLANKNO);
IF M >= 10 THEN
WRITELN(OUTFILE,"AVE",M:2,":=SUM",M:2,"/ACC;")
ELSE WRITELN(OUTFILE,"AVE",M:1,":=SUM",M:1,"/ACC;")
END;
CODEWRITE
END
END
END
END;(* CODESTATE12 *)

PROCEDURE CODESTATE3;
BEGIN
WHILE NOT EOF(OUTF) DO
BEGIN
  READLN(OUTF,READEXP);
  PRINTBLANK(BLANKNO);
  PRINT601(READEXP);
  WRITELN(OUTFILE);
END;
CODERITE
END; (*CODESTATE3 *)

(*************MAIN PROGRAM*******************)
BEGIN (* MAIN PROGRAM *)
OPENFILE;
I:=O; R:=O; C:=O; CC:=O;
SOURCENO:=O; GV:=O; GO:=O;
DATANO:=O; BENO:=O;
RECORDNO:=O; OUTNO:=O;
CH:=CHR(ASP);
BLANKNO:=O;
REPEAT
  WRITELN; WRITELN;
  WRITELN(" MENU");
  WRITELN(" ="="="");
  WRITELN;
  WRITELN(" (1) INPUT DATA");
  WRITELN(" (2) DATA PROCESSING");
  WRITELN(" (3) OUTPUT DATA");
  WRITELN(" (4) QUIT");
  WRITELN;
  WRITELN("PLEASE CHOOSE A MENU OPTION BY THE NUMBERS.");
  WRITELN;
  READLN(MENUNO);
  CASE MENUNO OF
    1:SOURC;
    2:DATAPROCESSING;
    3:DESTINATION;
    4:GOTO 10
  END
UNTIL MENUNO=4;
WRITELN;

(*-----------------
THIS PART IS TO CODE THE PROGRAM HEADING,*)
10:IF (SOURCENO <> 0) OR (DATANO <> 0) OR (OUTNO <> 0) THEN
BEGIN
  WRITELN;
  WRITELN("PLEASE ENTER THE DESIRED PROGRAM NAME (<30 CHAR)?");
  WRITELN;
  READLN(PROGNAME);
  WRITE(OUTFILE,"PROGRAM ");
  PRINT30(PROGNAME);
  IF SOURCENO=0 THEN
    WRITE(OUTFILE,"(OUTPUT")
  ELSE WRITE(OUTFILE,"(INPUT,OUTPUT")
  IF SOURCENO=1 THEN
    IF OUTNO=1 THEN
      WRITELN(OUTFILE,"INFILE,OUTFILE")
    ELSE WRITELN(OUTFILE,"INFILE")
  END
ELSE BEGIN
  IF OUTNO=1 THEN

DECLARE VARIABLE;

IF DATANO <> 0 THEN
  IF DATANO=1 THEN
    BEGIN
      IF (RECORDNO > 0) OR (SOURCENO=0) THEN
        WRITELN(OUTFILE, 'J:INTEGER; (* J IS FOR-LOOP INDEX.*)')
    END;
  ELSE BEGIN
    IF GV <> 0 THEN
      BEGIN
        WRITE(OUTFILE, 'BEGIN');
        FOR II:=1 TO GV DO BEGIN
          IF II >= 10 THEN
            WRITE(OUTFILE, 'SUM'; II:2)
          ELSE WRITE(OUTFILE, 'SUM'; II:1);
          IF II <> GV THEN
            WRITE(OUTFILE, ',');
        END;
      IF BENO=2 THEN
        BEGIN
          WRITE(OUTFILE, 'END');
          WRITELN(OUTFILE, ';REAL;');
        END;
      END;
      IF BENO=2 THEN
        WRITE(OUTFILE, 'ACC;REAL;')
    END;
  END;
  ELSE IF SOURCENO <> 0 THEN
    BEGIN
      IF SOURCENO=1 THEN
        BEGIN
          IF OUTNO=1 THEN
            WRITE(OUTFILE, 'BEGIN');
          ELSE WRITE(OUTFILE, 'INFILE:TEXT;')
        END;
    ELSE IF OUTNO=1 THEN
      WRITE(OUTFILE, 'OUTFILE:TEXT;')
    END;

    WRITELN(OUTFILE);
    (*-----------------------------------------------*)
    HERE BEGIN TO CODE MAIN PROGRAM. (*)
    WRITELN(OUTFILE, 'BEGIN');
    IF SOURCENO <> 0 THEN
      BEGIN
        IF SOURCENO=1 THEN
          BEGIN
            IF OUTNO=1 THEN
              WRITE(OUTFILE, 'BEGIN');
          END;
        END;
    END;
BEGIN
  OPENIN;
  OPENOUT
END
ELSE OPENIN
END
ELSE IF OUTINO=1 THEN
  OPENOUT
END;
CLOSE(OUTF);
OPEN(OUTF, "F1.DAT", HISTORY: = OLd);
RESET(OUTF);
BLANKNO:=BLANKNO + 3;
CASE SOURCENO OF
  0,1,2: CODESTATE12;
  3: CODESTATE3
END;
BLANKNO:=BLANKNO - 3;
PRINTBLANK (BLANKNO);
WRITELN (OUTFILE, "END.");
WRITELN; WRITELN;
(*-----------------TAKE SOME MESSAGES TO USER, --------------------------*)
WRITELN( "NOW, YOUR PROGRAM IS COMPLETED,");
IF SOURCENO=1 THEN
BEGIN
  WRITELN;
  WRITELN("NOTICE: ");
  WRITELN("CREATE AN INPUT FILE BEFORE YOU RUN THIS PROGRAM.");
  WRITE("YOUR FILE NAME IS: ");
  II:=1;
  WHILE FILENAME1[II] <> '' DO
  BEGIN
    WRITE(Filename1[II]);
    II:=II + 1
  END;
  WRITELN; WRITELN
END;
WRITELN("SEE YOUR PROGRAM, TYPE "T F2.PAS ");
WRITELN("FOLLOWING THE "s" SIGN.");
WRITELN; WRITELN
END
END, (* MAIN PROGRAM *)
Appendix B

Running processing and output
of the LAST.PAS program
$RUN LAST

MENU
=====
(1) INPUT DATA
(2) DATA PROCESSING
(3) OUTPUT DATA
(4) QUIT

PLEASE CHOOSE A MENU OPTION BY THE NUMBERS.

1

THE SOURCE OF INPUT:
(1) FILE
(2) TERMINAL
(3) IN PROGRAM

PLEASE CHOOSE ONE OF NUMBERS.

2

INPUT DATA MAP:

-------------------------------------------------------------
THIS DATA MAP IS THE DESCRIPTION OF INPUT DATA.
IT INCLUDES TWO PARTS:
(1) DATA TYPE: THERE ARE THREE DATA TYPES: INTEGER, REAL
AND CHARACTER. TYPE I, R, OR C TO REPRESENT THEM.
(2) COLUMN LOCATION: INPUT DATA FORMAT.
-------------------------------------------------------------

TYPE "Q", IF FINISH.

INPUT DATA MAP
==================

<table>
<thead>
<tr>
<th>DATA TYPE</th>
<th>COLUMN LOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>1-2</td>
</tr>
<tr>
<td>I</td>
<td>4-5</td>
</tr>
<tr>
<td>R</td>
<td>7-9</td>
</tr>
<tr>
<td>Q</td>
<td></td>
</tr>
</tbody>
</table>

YOU HAVE FOLLOWING VARIABLES:
II, I2, R1

HOW MANY RECORDS DO YOU HAVE?

5
(1) INPUT DATA
(2) DATA PROCESSING
(3) OUTPUT DATA
(4) QUIT

PLEASE CHOOSE A MENU OPTION BY THE NUMBERS.

2

COMPUTE
(1) IN SAME RECORD.
(2) BETWEEN RECORDS.

PLEASE CHOOSE ONE OF NUMBERS.

1

DO YOU NEED LOOP IN PROGRAM? (Y OR N)
Y
J IS LOOP INDEX.

WITHIN LOOP FORMULA(S):
NOTICE:
(1) EACH FORMULA CAN NOT EXCEED 50 CHARACTERS.
(2) ONE LINE FOR ONE FORMULA.
(3) TYPE 'Q', IF FINISH.

(I1+I2)*R1
R2 IS THE VARIABLE OF THIS FORMULA.

Q

(1) INPUT DATA
(2) DATA PROCESSING
(3) OUTPUT DATA
(4) QUIT

PLEASE CHOOSE A MENU OPTION BY THE NUMBERS.

3

THE DESTINATION OF OUTPUT:
(1) FILE
(2) TERMINAL

PLEASE CHOOSE ONE OF NUMBERS.
OUTPUT DATA MAP:

-----------------------------------
THIS DATA MAP IS THE DESCRIPTION OF OUTPUT FORMAT.
IT INCLUDES TWO PARTS:
(1) VARIABLE OR STRING: IF VARIABLE, TYPE I1, R1, OR C1
   TO REPRESENT THE FIRST INTEGER, REAL, OR CHAR.
   IF STRING, USE A QUOTATION MARK TO COVER IT.
(2) SPACE: SPACE NUMBERS FOR THIS VARIABLE OR STRING.
   IF REAL, TYPE BOTH WHOLE SPACE NUMBERS AND THE
   SPACE NUMBER AFTER DECIMAL POINT; USE A COLON TO
   SEPARATE THEM.
-----------------------------------

TYPE 'Q', IF FINISH.

OUTPUT DATA MAP

-------------------------------
VARIABLE OR STRING    SPACE
-----------------------
R2                     7:1
Q

MENU
=====
(1) INPUT DATA
(2) DATA PROCESSING
(3) OUTPUT DATA
(4) QUIT

PLEASE CHOOSE A MENU OPTION BY THE NUMBERS.

4

PLEASE ENTER THE DESIRED PROGRAM NAME (<=30 CHAR) ?

TEST

NOW, YOUR PROGRAM IS COMPLETED.
SEE YOUR PROGRAM, TYPE 'T F2.PAS'
FOLLOWING THE '$' SIGN.

$ T F2.PAS
PROGRAM TEST INPUT, OUTPUT;
VAR
   I1, I2: INTEGER;
   R1, R2: REAL;
   J: INTEGER; (* J IS FOR-LOOP INDEX.*)
BEGIN
   WRITELN('PLEASE ENTER THE DATA.');
   FOR J := 1 TO 5 DO
   BEGIN
      READLN(I1, I2, R1);
      R2 := (I1 + I2) * R1;
      WRITELN(R2: 7: 1);
   END;
END.
$PAS F2
$LINK F2
$RUN F2
PLEASE ENTER THE DATA.
25 31 5.6
   313.6
12 90 4.1
   418.2
10 29 3.1
   120.9
50 24 7.2
   532.8
30 51 3.9
   315.9
$
$ RUN LAST

MENU
======

(1) INPUT DATA
(2) DATA PROCESSING
(3) OUTPUT DATA
(4) QUIT

PLEASE CHOOSE A MENU OPTION BY THE NUMBERS.

2

DO YOU NEED LOOP IN PROGRAM? (Y OR N)
Y
J IS LOOP INDEX,
WHAT IS MAXIMUM VALUE OF J ?
10

WITHIN LOOP FORMULA(S):
NOTICE:
(1) EACH FORMULA CAN NOT EXCEED 50 CHARACTERS,
(2) ONE LINE FOR ONE FORMULA,
(3) TYPE 'Q', IF FINISH.

\sqrt(J)
R1 IS THE VARIABLE OF THIS FORMULA.
Q

MENU
======

(1) INPUT DATA
(2) DATA PROCESSING
(3) OUTPUT DATA
(4) QUIT

PLEASE CHOOSE A MENU OPTION BY THE NUMBERS.

3

THE DESTINATION OF OUTPUT:
(1) FILE
(2) TERMINAL

PLEASE CHOOSE ONE OF NUMBERS.

2
OUTPUT DATA MAP:

--------------------------------------------------------------------------------------------------------------------------

THIS DATA MAP IS THE DESCRIPTION OF OUTPUT FORMAT.
IT INCLUDES TWO PARTS:
(1) VARIABLE OR STRING: IF VARIABLE, TYPE I1, R1, OR C1
    TO REPRESENT THE FIRST INTEGER, REAL, OR CHAR.
    IF STRING, USE A QUOTATION MARK TO COVER IT.
(2) SPACE: SPACE NUMBERS FOR THIS VARIABLE OR STRING.
    IF REAL, TYPE BOTH WHOLE SPACE NUMBERS AND THE
    SPACE NUMBER AFTER DECIMAL POINT; USE A COLON TO
    SEPARATE THEM.

--------------------------------------------------------------------------------------------------------------------------

TYPE 'Q', IF FINISH.

OUTPUT DATA MAP

==================

VARIABLE OR STRING   SPACE
------------------------------
J                      6
------------------------------
R1                     10:1
------------------------------
Q

MENU

=====

(1) INPUT DATA
(2) DATA PROCESSING
(3) OUTPUT DATA
(4) QUIT

PLEASE CHOOSE A MENU OPTION BY THE NUMBERS.

4

PLEASE ENTER THE DESIRED PROGRAM NAME (<=30 CHAR) ?

TEST1

NOW, YOUR PROGRAM IS COMPLETED.
SEE YOUR PROGRAM, TYPE 'T F2.PAS'
FOLLOWING THE '$' SIGN.

$ T F2.PAS
PROGRAM TEST1(OUTPUT);
VAR
  R1:REAL;
  J:INTEGER; (* J IS FOR-LOOP INDEX.*)
BEGIN
  FOR J:=1 TO 10 DO
  BEGIN
    R1:=SQRT(J);
    WRITELN(J:6,R1:10:1);
  END;
END.
$PASF2
$LINKF2
$RUNF2

1       1.0
2       1.4
3       1.7
4       2.0
5       2.2
6       2.4
7       2.6
8       2.8
9       3.0
10      3.2
$