“The hardest part of programming is now over. This is where the fun begins.”

S. Neebar
INTRODUCTION

The final phase in the development of a program is the Program Implementation phase. This phase involves:

Step 1: Translate the algorithm into a specific programming language.

Step 2: Execute the program on the computer.

Step 3: Maintain the program.
TRANSLATE THE ALGORITHM INTO A SPECIFIC PROGRAMMING LANGUAGE

The programming language to be used for this course is: Pascal...

So Why Pascal?
• Well firstly it is simple.
• It was designed by Professor Nicklaus Wirth in 1968.
• Specifically designed as a tool to teach programming to beginners.

So what makes Pascal perfect for beginners?

1. It is well-structured.
2. It is easy to implement.
3. The syntax (grammar) is easy to learn and follow.
4. It encourages the programmer to adopt a disciplined approach to programming.
BEFORE TRANSLATING OUR ALGORITHM INTO PASCAL...

• Firstly, we have to learn the structure and syntax of the language.

• Syntax, meaning, the rules of the language that govern the grammatical issues such as, the vocabulary, word placement and punctuation.

NOTE: Since Pascal is a language for communicating with computers, the rules are somewhat different from those of human interaction languages such as English or French.

Example: there is a limit on the number of characters that a Pascal word can have.
Variants of Pascal

Just as most languages have certain variants, called dialects, so does the Pascal language.
The version approved by the International Standard Organization (ISO) is referred to as Standard Pascal.

Other variants are:
- Turbo Pascal
- Think Pascal
A Pascal program has three distinct parts:

1. The program heading
2. The program block
3. The program terminator (a period).
**Program Heading**

*Program heading* is a single statement beginning with the word *program*.

The heading assigns a name to the program and lists the input and output streams in parentheses.

The program block is the body of the program. It consists of the Pascal statements for executing the algorithm.

The block is divided into two distinct parts:

1. The *variable declaration section* where all the variables and data structures used by the program are defined.
2. The statement section is where all the action statements of the program are specified. The statement section is encapsulated within begin and end statements.
**Begin & End Keywords**

Begin and end are examples of keywords used in Pascal. Keywords (or reserved words) are words that have special meaning in Pascal and can only be used in the predefined context. That is, they cannot be used as variable names or in any other context.

Other keywords are:
- Program
- type
- var
- const
- read
- write
- read In
- write In
Pascal Program Template

Program name (input, output) ;

Definition/Variable declaration section:

label declarations
const definitions
type definitions

Procedure/ function declaration
{Main Program}
Begin

statement;
statement;
statement;

End.
Pascal Syntax in a Nutshell

1. Declaring Variables

2. Data Types

3. User Identifiers

4. Punctuation

5. The Assignment Symbol

6. Begin and End Delimiters

7. Comments
1. Declaring Variables

Variables must always be declared in the variable declaration section prior to their use in the program. Variables are declared in Pascal by specifying the keyword `var` followed by a list of variables, a semi-colon and the data type.

For example,

```pascal
var
  num1: integer;
  average : real;
```
2. **Data Types**

A data type is a collection of elements that are all formed and treated the same way.

For example:
- integers
- real numbers
- characters
- Boolean

Pascal uses the following keywords to identify the various data types:

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Keyword</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer</td>
<td>integer</td>
</tr>
<tr>
<td>Real</td>
<td>real</td>
</tr>
<tr>
<td>Character</td>
<td>char</td>
</tr>
<tr>
<td>string</td>
<td>String*</td>
</tr>
<tr>
<td>Boolean</td>
<td>Boolean</td>
</tr>
</tbody>
</table>

*Non-Standard*
3. User Identifiers

These are names created by the programmer. These include variable names, the program name, names of symbolic constants or names of subprograms.

The Rules governing user identifiers in Standard Pascal are as follows:

• an identifier must start with an alphabetic character (upper or lower case)
• it can be composed of only alphabetic characters or a mixture of alphabetic and numeric characters, commonly called alphanumeric.
• No special characters are allowed (like &, + -). Some variants of Pascal allow the underscore character.
• The identifier may be of any length, but standard Pascal only recognizes the first 8 characters to determine uniqueness. Examples of valid user identifiers are:
  - temp
  - value1
  - Number of Items
  - X
  - X1
• Reserved words or Pascal keywords cannot be used as identifiers.
• Pascal is NOT case-sensitive. Therefore, there is no distinction between the identifiers Number and number, for example. They are considered to be the same.
4. PUNCTUATION

Every Pascal statement (except begin) is terminated by a semi-colon. The last statement in the program (that is, the end statement) is terminated by a period.

5. The Assignment Symbol

In Pascal the assignment symbol is “:=“. This corresponds to the “←“ in our pseudo code.
6. **Begin and End Delimiters**

The begin/end keyword pair is used to delimit the body of the program as well as logical blocks within the program. For example, when multiple statements are to be executed within a while loop, such as statements are encapsulated within a begin/end pair. For every begin in an program, there must be a corresponding end statement.

7. Comments

Comments are used to document a program internally, to aid in the understanding of the program statements and segments. Comments in Pascal are enclosed within curly braces `{....}`. Comments are ignored by the Pascal compiler. That is, they do not affect the logic of the program or the syntax in any way. They are only there to allow the programmer to make notifications about the code.

The best way to learn any programming language is to look at a simple program and note the common features. Let us take one of the algorithms we developed in previous chapters and see how it translates into Pascal.
TRANSLATING PSEUDOCODE INTO PASCAL CODE

Step 1: make a list of all the variables used in the algorithms and determine their type. That is, the type of values that each variable is to store.

In the algorithms, we were not too concerned about the types of variables we used, so we did not explicitly declare them in the algorithm. However, most programming languages require that variables be declared explicitly before they are used.

Step 2: translate each statement in the algorithm into its Pascal equivalent.
TRANSLATION OF THE AVERAGE PROBLEM

The first step is to list all the variables and their types. There are three variables in this algorithm, num1, num2 and num3. They are all of integer type.

Pascal Code:

Program Average (input, output);
{This algorithm finds the average of three numbers}
Var
  num1, num2, num3: integer
Begin
  readln(num1, num2, num3);
  average := (num1 + num2 + num3) / 3;
  writeln (‘The average is: ‘, average);
End.

Pseudocode:

This algorithm finds the average of three numbers

Start
Get num1, num2, num3
Average ← (num1 + num2 + num3) ÷ 3
Display “The average is”, average
Stop.
Q: What is the format of the header in Pascal?
A: Every Pascal program begins with the keyword program, followed by the name of the program and the input/output streams in parentheses. Our algorithm is called Average, so we can use the same name for the program to get:

Program Average (input, output);

A glance at the Pascal template will show that the next step is to declare our variables. In our algorithm, we did not declare all the variables. However, we must do so in the Pascal program.

Q: What is the equivalent of ‘start’ in Pascal?
A: Begin. So we write begin in our program.

Q: How do we input (read) values in Pascal?
A: Use read or readln, so we look up the format of the read/readln statement and write that in our program.
PASCAL Q & A...

Q: How do we perform arithmetic operations in Pascal?
A: The arithmetic operators in Pascal are pretty similar to those in mathematics with a few exceptions—multiplication is the asterisk (*) and division is the slash (/).

Q: What is the assignment symbol in Pascal?
A: The symbol is “:=” So now all we have to do is rewrite the assignment statement, using the Pascal constructs. That is, average := (num1+num2+num3)/3;

Q: How do we display results/messages in Pascal?
A: we use the write or writeln statement.

Q: How do we end a Pascal program?
A: by the keyword, “end” followed by a period.
EXAMPLE 2:

Pascal Code:
Program Large Count (input, output);
Var
Count, NoOfValues : integer;
First Number, nextNumber: integer
Begin
count:= 0;
NoOfValues := 1;
while (NoOfValues <=20) do begin
  read1n (nextNumber);
  NoOfValues := No Of Values + 1;
  if (nextNumber > firstNumber) then
    count := count + 1;
end;
{end of while loop}
Write1n (firstNumber, count);
End.

Pseudocode:
Algorithm Large_Count

Given a list of 20 integers, this algorithm counts the number of values that are larger than the first value.
Set count to 0
Set No-of-Vales to 1
Get first-number
While (No-Of-Values <= 20 ) do the following:
  get next-number
  increment No-Of-Values
  if next-number > first-number then
    increment count
  end-if
End-while
Print first-number, count
Stop.
Points to Note in the Previous Example:

- The name of the program does not include the underscore character, as this is illegal in Standard Pascal.

- Multiple statements are to be executed within the while loop. This is referred to as a compound statement. Compound statements in Pascal are encapsulated within begin/end pairs.

- The increment count statement in the algorithm translates into count:= count + 1; in Pascal, likewise for increment No-Of-Values.

- The end statement, which marks the end of the while loop, is terminated by a semi-colon.

- The statement which ends the program is terminated by a period.
**A More Complex Example**

The next algorithm, exemplifies a complex control structure called the nested If structure. Nested control structures are outside the scope of this course. However, the algorithm demonstrates the importance of organizing the appearance of the statements to reflect the logic. Every if must have a corresponding else and the scope of the statements must be clearly delineated.
This algorithm reads test scores and determines the letter grade for each score.

Read test-score
While (test-score ≠ -1) do
    If (test-score > 100) OR (test-score < 0) then  {check is the data is valid}
        Print “Error: Invalid test score”
    else
        If (test-score >=80 and test-score <=100) then
            Lettergrade ← ‘A’
        Else
            If (test-score < 80 and test-score >=70) then
                Lettergrade ← ‘B’
            Else
                If (test-score <70 and test-score >=60) then
                    Lettergrade ← ‘C’
                Else
                    Lettergrade ← ‘D’
        End-If  {if test score is invalid}
    End-If
    Print test-score, lettergrade
End-If
Read test-score
End-while
Stop.
PASCAL IMPLEMENTATION:

program Grades (input, output);
Var
test-score: integer;
lettergrade: char;
Begin
Readln (test-score);
While (test-score <> -1) do
Begin
{Start of While loop}
if (test-score > 100) or (test-score < 0) then
writeln (‘Error: Invalid test score’);
else
begin
if (test-score >= 80) and (test-score <= 100)
Lettergrade := ‘A’;
else
if (test-score < 80) and (test-score >= 70)
Lettergrade := ‘B’;
else
Lettergrade := ‘C’;
else
Lettergrade := ‘D’
Writeln (test-score, lettergrade)
End; {if test score is invalid}
Readln (test-score);
End; {end of while loop}
End. {end program}
POINTS TO NOTE IN THE PREVIOUS EXAMPLE

- In the nested if statements all the matching else’s are in line with the corresponding ifs.
- The variable testscore is not hyphenated as in the algorithm.
- Compound statements are enclosed within begin/end blocks.
- The symbol for “not equal” is “<>”
- The format of the if and while statements closely resembles that of the algorithms. That’s okay. This format is used by many other languages, including English; it is not peculiar to Pascal.
SUMMARY

The above treatment of the Pascal programming language is not intended to be viewed as “all one needs to know about the programming language’. This was a mere overview of the basic features of the language to facilitate an illustration of the algorithm translation process.