CHAPTER 6

CONTROL STRUCTURES
PART 1 - SELECTION

Control Structure Part1 - Selection

6.1 Introduction

Control structures are used to control program flow. The three types of control structures that exist for program development are:

- Sequence
- Selection
- Iteration

A program may utilize a combination of the three available control structures to solve a problem. The control structure or structures used depends on the complexity of the problem being solved.

6.2 Sequence

Sequence control structure orders program instruction in a linear fashion. The program instructions are performed one after the other beginning with the first instruction and ending with the last instruction. All instructions are executed; none is skipped. All the programs we encountered in this text so far are sequential programs.

```
Step 1  input numb1
Step 2  input numb2
Step 3  Sum = numb 1+ numb2
Step 4  Average = sum/2
Step 5  print average
```

***Figure 6.1a** - Five-instructions sequence pseudocode*

The examples in Figure 6.1a & 6.1b have a five instructions sequence. Steps 3, 4 and 5 are dependent on steps 1 and 2. Step 5 is dependent on step 4. Sequence is the simpliest form of control structures in program development but as the problem becomes more complexed other control structures become necessary.

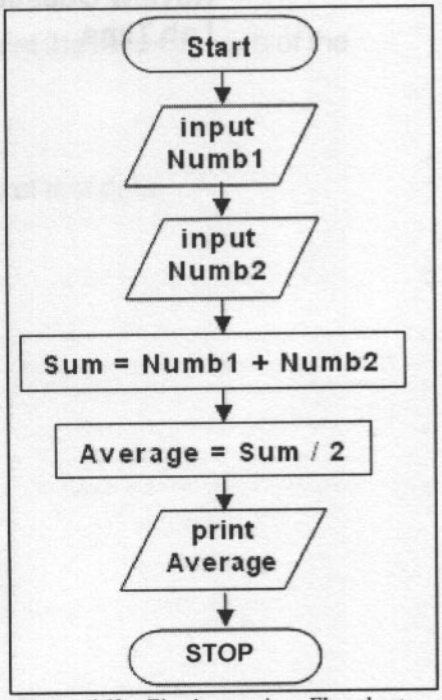


***Figure 6.1b** - Five instructions Flowchart sequence*

6.3 Selection

Selection control structures are used to make decisions. Consider the following two examples:

1. *If IT IS RAINING then TAKE A RAINCOAT*
2. *If IT IS RAINING then TAKE A RAINCOAT else TAKE AN UMBRELLA*

In both examples the condition is:

IT IS RAINING

This simple condition can either be true or false. For Example 1, if the condition is true then the decision. **TAKE A RAINCOAT**, is taken. If the condition is false, then no decision is taken because none is dictated.

For Example 2, if the condition is true then the decison **TAKE A RAINCOAT** is performed. If the condition is false then the decision **TAKE AN UMBRELLA** is carried out.

6.3.1 Condition

A **condition** is a boolean expression that evaluates to either a TRUE or a FALSE. The following are examples of boolean expressions:-

It is raining
Hungry and Thirsty
Age = 17
Age > 17 and Age < 20
23 + 30 = 49
"Isaac Newton" was a man

Each statement can either be true or false and can lead to a particular action being taken dependent on the truth value of the condition.

6.4 Pascal Selection Syntax

The syntax diagram for a Pascal Selection (IF) statement is:

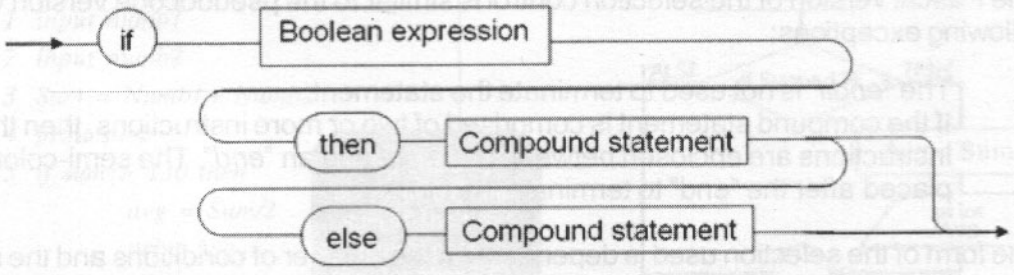


***Figure 6.2** - Syntax for Pascal selection control structure*

A compound statement represents one or more Pascal statements. When a compound statement is more than one statement, the statements are enclosed between a "begin" and an "end". The end in this case must be followed by a semi-colon (;).

6.4.1 PSEUDOCODE VERSION - SELECTION

```
If condition then
        compound statement
endif
```

Figure 6.3 - A single outcome selection structure

```
if condition then
        compound statement1
else
    c   ompound statement 2
endif
```

Figure 6.4 - Two outcomes selection structure

```
if condition1 then
        compound statement 1
else if condition 2 then
        compound statement2
else if condition 3 then
        compound statement 3
else if
        ,,      ,,
        ,,      ,,
else
    c   ompound statement n
endif
```

Figure 6.5 - Multiple outcomes selection structure derived from Figure 6.3 and 6.4

6.4.2 FLOW CHART VERSION - SELECTION

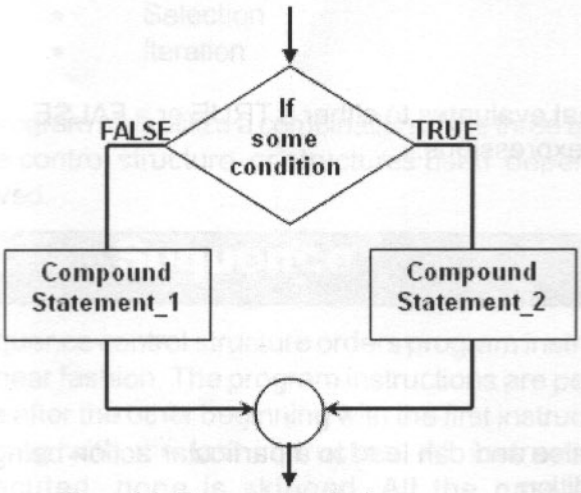


Figure 6.6 - Two outcomes selection structure

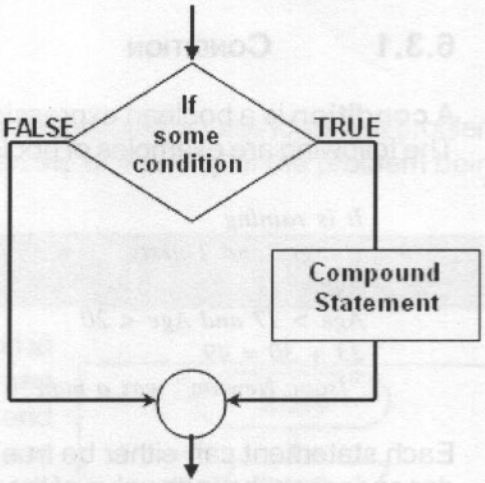


Figure 6.7 - A single outcome selection

6.4.3 PASCAL VERSION - SELECTION

The Pascal version of the selection control is similar to the pseudocode version with the following exceptions:

- The *"endif"* is not used to terminate the statement.
- If the compund statement is comprised of two or more instructions, then the instructions are enclosed between a *"begin"* and an *"end"*. The semi-colon is placed after the "end" to terminate the block.

The form of the selection used is dependent on the number of conditions and the number of outcomes the problem requires. If it is a single outcome then the form in Figure 6.1 can be used. If it is two possible outcomes then the form in Figure 6.2 can be used. If there exists more than two outcomes, then the form in Figure 6.3 can be used. A combination of all the forms can be used depending on the style and skill of the programmer.

PRACTICE EXAMPLE 17

Design pseudocode and flowchart to read two numbers. If the sum of the two numbers is greater than 130 then the program should also calculate and print the average of the three numbers.

SOLUTION

DISCUSSION

We need a boolean expression that tests whether the sum of the numbers is greater than 130. If we store the sum of the two numbers in a variable named Sum, then our boolean expression is:

sum > 130

We then need to test this expression. If it is true then we need to calculate the average of the three numbers. We only have one decision to make. We therefore need a single outcome conditional construct.

IPO Chart

Input	Process	Output
Numb1 Numb2	input Numb1 input Numb2 Sum = Numb1 + Numb2 Avg = Sum/2 print Sum print Average	Sum Avg

Figure 6.8 - IPO Chart for practice example 17

Control structures required:

We need a selection control structure with one outcome to test the condition ***Sum > 130.***

Pseudocode

```
Step 1  input numb1
Step 2  input numb2
Step 3  Sum = Numb1+ Numb2
Step 4  print sum
Step 5  if sum > 130 then
                avg = Sum/2
                print Avg.
        endif
```

Step 5 is only performed if the sum of the two numbers is greater than 130

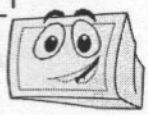


Figure 6.9 - Pseudocde for practice example 17

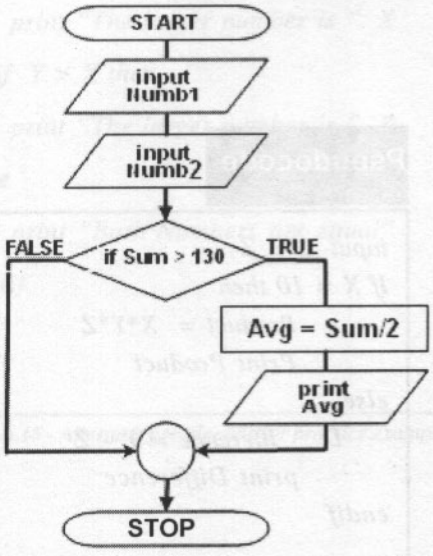


Figure 6.10 - Flowchart for practice example 17

PRACTICE EXAMPLE 18

Design pseudocode and flowchart to read three numbers. If the first number is 10 then the pseudocode should calculate and print the product of the numbers, otherwise it should calculate and print the difference of the second and third numbers.

SOLUTION

DISCUSSION

This problem requires two possible outcomes:

1. If the first number entered is 10 then we should calculate and print the product of the three numbers.
2. If the first number is not 10, then we should calculate and print the difference between the second and third numbers.

We therefore need a selection structure that provides us with two possible outcomes.

IPO Chart

Input	Process	Output
X Y Z	input X, Y, Z Product = X * Y * Z Difference = Y – Z print Product print Difference	Product Difference

Figure 6.11 - IPO Chart for practice example 18

Control structures required:

We need a selection control structure with two possible outcomes to test the condition **X = 10.**

Pseudocode

```
input X, Y, Z
If X = 10 then
        Product =  X*Y*Z
        Print Product
else
    D    ifference = Y – Z
        print Difference
endif
```

Figure 6.12 - Pseudocode for practice example 18

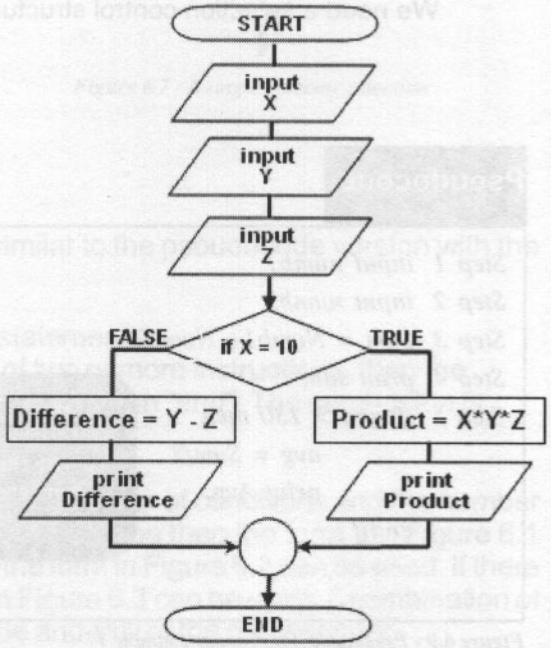


Figure 6.13 - Flowchart for practice example 18

PRACTICE EXAMPLE 19

Design pseudocode and flowchart that reads two values, determines the largest value and prints the largest value with an identifying message.

SOLUTION

DISCUSSION

Given two numbers, x and y, there are three possibilities:

1. The numbers are equal
2. X is greater than Y
3. Y is greater than X

With this question, there are three possible outcomes:

1. If the numbers are equal, we print a message stating that the numbers are equal.
2. If X is greater than Y, we print a message stating that X is greater than Y.
3. If Y is greater than X, we print a message stating that Y is greater than X.

Pseudocode 1

```
input X, Y

if X = Y then
    print "both numbers are equal"
endif

if X > Y then
    print "The greater number is ", X
endif

if Y > X then
    print "The greater number is ", Y
endif
```

***Figure 6.14** - Pseudocode for practice example 19*

Pseudocode 2

```
input X, Y

if X > Y then
    print "The larger number is ", X
elsif Y > X then
    print "The larger number is ", Y
else
    print "Both Numbers are equal"
endif
```

***Figure 6.15** - Alternate pseudocode for practice example 19*

Figure 6.16 - Flowchart solution corresponding to Figure 6.14 using multiple single outcome if constructs

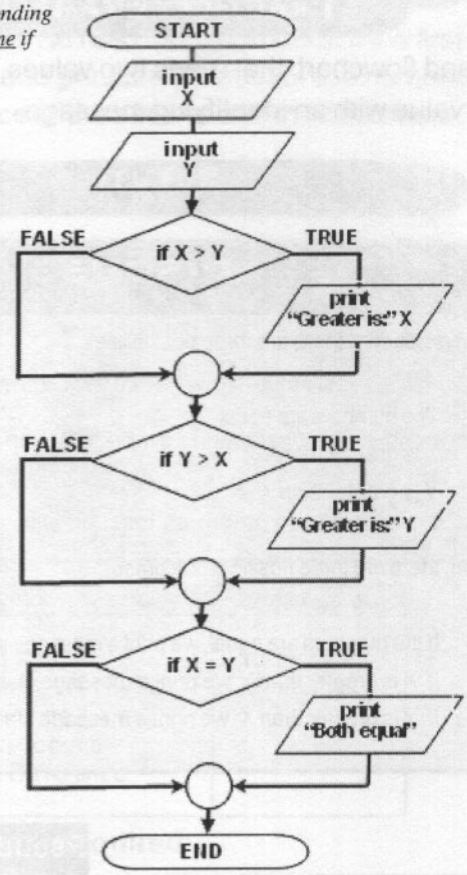
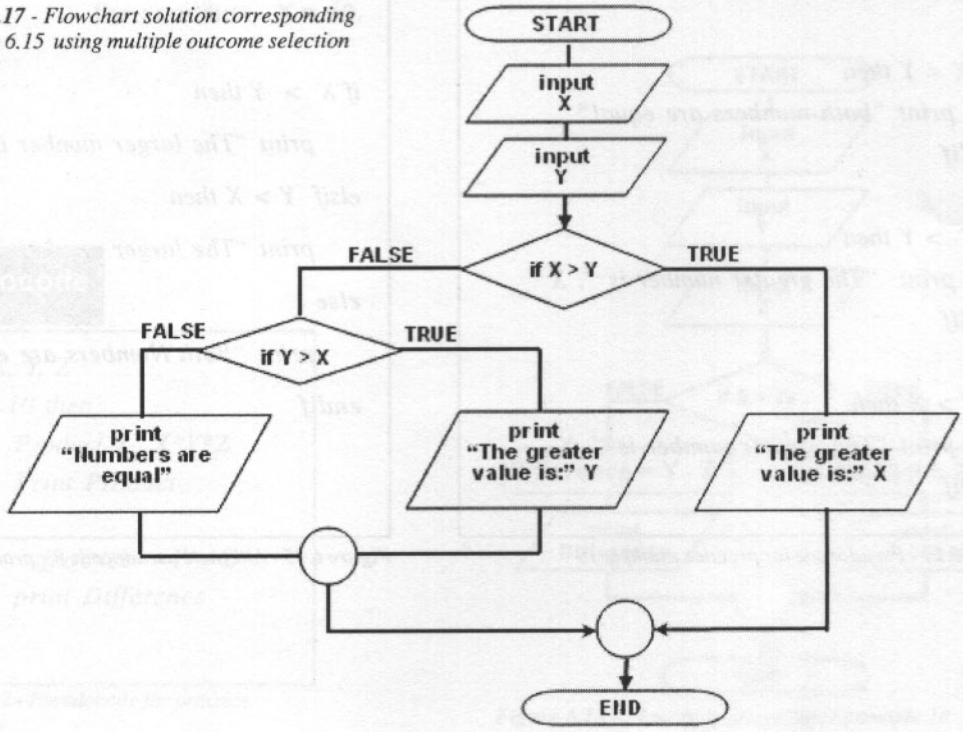


Figure 6.17 - Flowchart solution corresponding to Figure 6.15 using multiple outcome selection structure



PRACTICE EXAMPLE 20

Write pseudocode and Pascal code to accept input of an employee's name, overtime hours worked, hours absent and determine the employees bonus based on the bonus table below:

Bonus Rate Table	
If Indicator is:	Bonus Paid
40 hours or more	$5,000.00
More than 30 hours but less than or equal to 40 hours	$4,000.00
More than 20 hours but less than or equal to 30 hours	$3,000.00
More than 10 hours but less than or equal to 20 hours	$2,000.00
10 hours or less	$1,000.00

The Indicator is calculated using the formula:

*Indicator = Overtime hours worked - (2/3)*hours absent*

SOLUTION

DISCUSSION

This problem requires a selection structure that provides us with five possible outcomes. We could use five single outcome constructs (Figure 6.1) or a multple outcome construct (Figure 6.2).

IPO Chart

Input	Process	Output
EmployeeName OvertimeHours HoursAbsent	input EmployeeName input OvertimeHours input HoursAbsent Indicator = OverTimeHours – 2/3*HoursAbsent print BonusPaid	BonusPaid

Figure 6.18 - IPO Chart for practice example 20

Control structure needed

We need a selection structure to test ***indicator*** and output the appropriate bonus value.

Pseudocode

```
input EmpName
input OvertimeHours
input HoursAbsent

Indicator = OverTimeHours – 2/3*HoursAbsent

If Indicator > 40 then
        BonusPaid = 5000.00
endif
if Indicator >30 and indicator <= 40 then
        BonusPaid = 4000.00
endif
if Indicator >20 and indicator <= 30 then
        BonusPaid – 3000.00
endif
if Indicator >10 and indicator <= 20 then
        BonusPaid = 2000.00
endif
if Indicate <= 10 then
        BonusPaid = 1000.00
endif

print BonusPaid
```

Figure 6.19 - Pseudocode for practice example 20

Pascal Code

```
Program Bonus (input, output);
var
    indicator, OvertimeHours: real;
    HoursAbsent, BonusPaid: Real;
    EmpName: String;
begin
    write ('Enter the employee name -:');
    readln (EmpName);
    write ('Enter Overtime Hours For This Employee -:');
    readln(OverTimeHours);
    write('Enter Hours Absent for This Employee -:');
    readln(HoursAbsent);
    indicator := OvertimeHours - (2/3)*HoursAbsent;

    If indicator > 40 then
        BonusPaid := 5000.00;
    If (indicator > 30) and (indicator <= 40) then
        BonusPaid := 4000.00;
    If (indicator > 20) and (indicator <= 30) then
        BonusPaid := 3000.00;
    If (indicator > 10) and (indicator <= 20) then
        BonusPaid := 2000.00;
    If indicator <=10 then
        BonusPaid := 1000.00;

    writeln(EmpName, ' Bonus paid is : ',BonusPaid:3:2);
    readln;
end..
```

Figure 6.20 - Pascal code for practice example 20

SOLUTION

Write pseudocode and Pascal code to read three numbers and print the value of the smallest number. The program should also indicate if all three numbers are equal.

PRACTICE EXAMPLE 21

DISCUSSION

One method to solve this problem is to work with the first two numbers finding the smallest and storing it in a fourth variable. Finally, we work with the third variable and the fourth variable to find the smallest. A problem can have several different solutions. The programmer must assess each solution and choose the best one.

IPO Chart

Input	Process	Output
X Y Z	input X input Y input Z Smallest = X Smallest = Y Smallest = Z	Smallest

Figure 6.21 - IPO Chart for practice example 21

2. Control structure(s) required:

We need selection structures to compare X, Y and Z to determine the smallest value.

Pseudocode

```
input X, Y, Z

{working with the first two numbers}
If X < Y then
        Smallest = X
else
        Smallest = Y
endif

{working with the last number and smallest}
if Z < smallest then
        Smallest = Z
endif

{testing for equality or printing the smallest}
if ((X = Y) and (Y = Z)) then
        print "all three numbers are equal"
else
        print "Smallest number is ",Smallest
endif
```

Figure 6.22 - Pseudocode for practice example 21

Pascal Code

```
program smallest_Number (input, output);
var
    X, Y, Z, Smallest: real;

begin
    write('Enter three numbers -: ');
    readln(X, Y, Z);

    {**Working with the first two numbers **}
    if X < Y then
        Smallest := X
    else
        Smallest := Y;

    {**Working with the last two numbers **}
    if Z < smallest then
        smallest := Z;

    {**testing for equality or printing the smallest **}
    if (X = Y) and (Y = Z) then
        writeln('all three numbers are equal')
    else
        writeln('Smallest number is ', Smallest:3:2);

    readln;
end.
```

Figure 6.22 - Pascal code for practice example 21

Review Questions

1. List the three types of control structures giving a brief explanation of any two.
2. Write two variations of the selection structure stating the condition under which each is used.
3. Consider the following question:

   Design a program to read an integer value for SCORE and print the appropriate grade based on the following table:

SCORE	GRADE
80 or more	A
Less than 80 but 65 or more	B
Less than 65 but 50 or more	C
Less than 50 but 40 or more	D
Less than 40	F

   List the five boolean expressions needed by this program.

4. What determines the type of selection structure used for decision-making?
5. List the boolean expression needed by the following program:

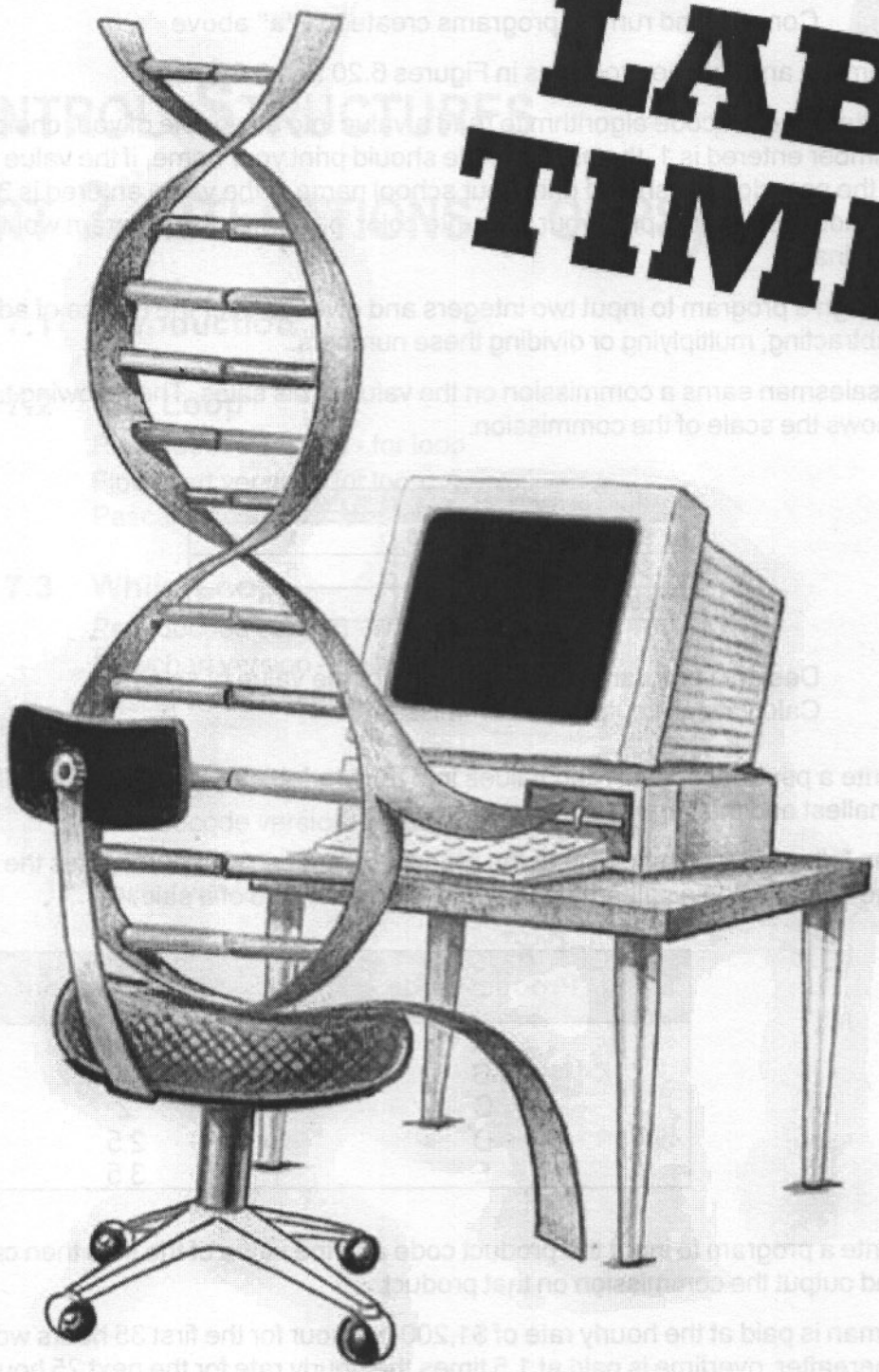
   (a) "Write an algorithm to read in TWO numbers into variables A and B. The algorithm should store the smaller in A and the larger in B, and then print A and B."

   (b) Write a program to read two numbers and divide the second number by the first number only if the first number is not equal to zero (0).

   (c) A concert organizer wants to charge different entrance prices as follows:

   Males over the age of fifteen pay $50. Females over fifteen years old pay $40. Female fifteen years old or younger must pay $20. Males fifteen and under pay $30. No person under 2 years old is allowed. Write a program or algorithm to read the name, age and sex of EACH patron. For EACH person, print the name and entrance fee. The program must stop when it encounters a person named "END."

LAB TIME



TRY THIS

1. (a) Convert the pseudocodes in Figures 6.9, 6.12, 6.14 and 6.15 to Pascal codes

   (b) Compile and run the programs created in **"a"** above

2. Compile and run the programs in Figures 6.20 and 6.22.

3 Write a pseudocode algorithm to read a value into a variable of your choice. If the number entered is 1, the pseudocode should print your name, if the value entered is 2, the pseudocode should print your school name, if the value entered is 3 the pseudocode should print your favourite color, otherwise the program would terminate.

4 Design a program to input two integers and give the user the choice of adding, subtracting, multiplying or dividing these numbers.

5. A salesman earns a commission on the value of his sales. The following table shows the scale of the commission.

Values of Sales			% Commission
$20,000	-	$39,000	1
$40,000	-	$99,000	5
$100,000	-	$500,000	10

   i) Design a program to input a figure for the value of sales
   ii) Calculate and output the commission

6 Write a pseudocode to read values into four variables A, B, C and D and print the smallest and the largest of the four variables.

7. The following table shows that the product code of an article indicates the percentage commission a salesman can earn from the value of a sale.

Product code	% commission
A	0.5
B	1
C	2
D	2.5
E	3.5

   Write a program to input the product code and the value of the sale then calculate and output the commission on that product.

8. A man is paid at the hourly rate of $1,200 per hour for the first 35 hours worked. Thereafter, overtime is paid at 1.5 times the hourly rate for the next 25 hours worked and 2 times the hourly rate for further hours worked. Write a program to input the number of hours worked per week, calculate and output the overtime paid.