9.1 Simple Selection

So far we have looked at two of the three basic constructs in computer programs. The first was a simple sequence of statements, executed one after another, and the second was a loop construct in which the body of the loop is executed repeatedly. In this chapter we look at the selection construct by which you can get the computer to select one of a number of alternative courses of action.

Here is a program that reads in the mark in a course and outputs a pass or fail standing. If the mark is 50 or more the computer will say that it is a pass; if it is less than 50 it will say that it is a failure.

```plaintext
% The "PassFail" program
% Read a mark and decide whether it is a pass or failure
put "Enter a mark " ..
var mark : int
get mark
if mark >= 50 then
   put mark, " is a pass"
else
   put mark, " is a failure"
end if
```

Remember the >= sign means is greater than or equal to. The selection construct is sometimes called the if...then...else construct. The statement (or statements) between the then and the else are executed if the condition after the if is true. They are called the then clause. If the condition is false, the statement (or statements) between the else and the end if are executed. These are called the else clause. Only one of the then or else clauses is executed. Which one is selected depends on whether the condition is true or false.
If there is nothing to be done when the condition is false, the keyword `else` and the `else` clause are omitted. For example, here is a program where the `if` statement does not have an `else` clause.

```plaintext
% The "FindLargest" program
% Read 10 positive integers and find the largest
put "Enter 10 positive integers"
var number : int
var largest := -1
for i : 1 .. 10
    get number
    assert number >= 0
    if number > largest then
        largest := number
    end if
end for
put "Largest integer of ten is ", largest
```

The variable `largest` is initialized to be \(-1\) which will be smaller than any of the positive integers entered. Its type is implied to be `int` since its initial value \(-1\) is an integer. Whenever a number that is bigger than the current value of `largest` is read, the value of the variable is changed to that new value. When the loop is finished the largest integer will have been stored in `largest`.

In the loop, after the `get` statement, there is a statement we have not seen before which says

```plaintext
assert number >= 0
```

If the condition after the keyword `assert` is false a run-time error will occur and execution of the program will be stopped. This prevents you from continuing if you enter a negative integer by mistake.
9.2 Three-way Selection

While simple selection allows a choice between two alternatives, often the solution to a given programming problem requires more choices. For example, determining the price of a movie ticket requires at least three alternatives based on the age of the movie-goer. Here is a program that calculates the price of a ticket based on age. This program calculates the ticket price for three alternatives: senior, adult, and child.

```turing
% The "MoviePrice" program
% Reads age and gives ticket price
var age : int
loop
    put "Please enter your age " ..
    get age
    assert age > 0
    if age >= 65 then
        put "Please pay $3.00"
        put "You are a senior"
    elsif age >= 14 then
        put "Please pay $6.00"
        put "You are an adult"
    else
        put "Please pay $2.50"
        put "You are a child"
    end if
end loop
```

This program will continue running until you interrupt it. The `assert` statement checks to see if the age is positive so that entering a negative age will stop it.

When an age is entered and has been passed by the `assert` statement, the program enters the three-way selection construct. This contains a new keyword `elsif`. If the condition `age >= 65` is true, the `then` clause is executed and the output is

```
Please pay $3.00
```
You are a senior

If the condition is false the person is either an adult or a child, and their age is less than 65. After the elsif, which is a special contraction of the two keywords else and if, we meet the second condition. If this condition is true, that is, age >= 14 the next two put statements are executed and the output is

Please pay $6.00
You are an adult

If the condition after the elsif is false, the statements after the else are executed with the output

Please pay $2.50
You are a child

Depending on the age of the person, one of the three possible alternatives is selected. We could call such a program construct an if...then...elsif...then...else construct. Like the if...then...else construct, the construct is terminated by the end if.

It is important to remember that the else must always come after the elsifs in a selection statement.

When using three-way selection, it is important to order the if statements correctly. Remember that as soon as the condition if tested true, then the execution goes to the end if. If we were to modify the program so that we switched the if’s as shown in the BadMoviePrice program

% The "BadMoviePrice" program.
...
  if age >= 14 then
    put "Please pay $6.00."
    put "You are an adult."
  elsif age >= 65 then
    put "Please pay $3.00."
    put "You are a senior."
  else
    put "Please pay $2.50."
    put "You are a child."
everyone would be adult or child. There would be no seniors. You must order from lowest to highest or highest to lowest.

It is not necessary to have an `else` clause. Here is a program that allows the user to buy three levels different products.

```turing
% The "Grocery" program
% Runs a cash register for a farmer's market stall selling a dozen corn
% for $2.25, bags of potatoes for $6.50, and artichokes for $1.75.
var total : real := 0
var product : int
loop
  put "Enter product (1=Corn, 2=Potatoes, 3=Artichokes, 4=Quit): " ..
  get product
  if product = 1 then
    total := total + 2.25
  elsif product = 2 then
    total := total + 6.50
  elsif product = 3 then
    total := total + 1.75
  elsif product = 4 then
    exit
  end if
% The phrase "total : 0 : 2" outputs total to two decimal places
put "The running total is $", total : 0 : 2
end loop
put "Final total = $", total : 0 : 2
```

9.3 Multi-way Selection

Here is a program that has more than three alternatives. It also uses a selection construct to test for improper data instead of using the `assert` condition. In this way you ask the person entering the data to correct it. This program reads a mark and gives the corresponding letter.
% The "Grade" program
% Read mark and find corresponding letter grade
put "Enter marks, end with 999"
var mark : int
loop
  put "Enter mark: " ..
  get mark
  exit when mark = 999
  if mark >= 0 and mark <= 100 then
    if mark >= 80 then
      put "A"
    elsif mark >= 70 then
      put "B"
    elsif mark >= 60 then
      put "C"
    elsif mark >= 50 then
      put "D"
    else
      put "Fail"
    end if
  end if
else
  put "Improper mark"
end if
end loop

The multi-way if selection construct is in the body of a conditional loop which starts with loop and finishes with end loop. The exit condition of the loop is true when you enter the dummy mark 999. After the first keyword if in the program is the compound condition

\[ mark \geq 0 \text{ and } mark \leq 100 \]

The and means that both these simple conditions must be true: the mark must be between 0 and 100 inclusive. If it is true the then clause is executed otherwise the else clause is executed, and the words ÖImproper markÖ will be output.

The then clause contains a selection construct, a multi-way one at that. This multi-way selection construct is nested inside the then clause of the two-way selection construct. The nesting is
emphasized by the indentation of the lines of the program. A multi-way selection has one or more `elsif...then` clauses. Only one of the `then`, `elsif...then`, or `else` clauses is executed. The multi-way selection construct, like the two-way selection construct, ends with the keywords `end if`.

Ordering of the if statements is also crucial in multi-selection. They must go from highest to lowest or from lowest to highest. Do not mix up the order of the conditions as it will influence the logic of the program. Examine the `BadGrade` program.

```plaintext
% The "BadGrade" program
... 
  if mark >= 60 then
    put "C"
  elsif mark >= 50 then
    put "D"
  elsif mark >= 80 then
    put "A"
  elsif mark >= 70 then
    put "B"
  else
    put "F"
  end if
... 
```

A mark of 99, 100, 75, 65, or 60 would all be Cs. This program would grant Ds to marks between 50 and 59 and Fs to all others.

9.4 Case Construct

When the choice among alternatives of a multi-way selection is determined by an integer value you can use the case construct. Here is a program that counts the number of votes for three political parties called Left, Middle, and Right. To vote for one of these enter a 1, or a 2, or a 3 respectively. To end the voting procedure enter −1.
% The "Voting" program
% Read votes and count totals
const sentinel := –1
put "Vote 1 for Left, 2 for Middle, 3 for Right, end with ", sentinel
var vote : int
var left, middle, right : int := 0  % initialize all three to 0
const leftVote := 1
const midVote := 2
const rightVote := 3
loop
  put "Enter vote ".
  get vote
case vote of
    label leftVote : left := left + 1
    label midVote : middle := middle + 1
    label rightVote : right := right + 1
    label sentinel : exit
    label : put "Invalid vote"
end case
end loop
put "Left" : 8, "Middle" : 8, "Right" : 8
put left : 4, middle : 10, right : 7

In the case statement, if the value of vote is 1 the statement following label leftVote: will be executed and 1 will be added to the total left. If the value of vote is the signal then the exit causes the loop to stop. If the value of vote is not any one of those specified after the label keywords then the label with no value following it will be selected. This one acts like an else clause in the if..then..else construct and must be the last label in the case statement.

Notice the rather peculiar field sizes for the integers in the last put statement. These will result in the lining up of the total votes in each of the three categories with the headings.

Remember strings are left-justified in their fields and integers (or reals) are right-justified. Try changing the size of each field to 8, the same as for the headings, and see the difference.
The **case** construct does not allow you to do anything that the **if..then..else** construct cannot do but sometimes it seems more understandable. It is also more efficient.

### 9.5 Commands for Action

One of the important things you can do with the selection construct is to issue commands to the computer to take different actions depending on what the command is.

Here is an example program which uses the selection construct.

```plaintext
% The "ShowEmotion" program
% Respond to various commands
var command : string
loop
  put "Enter command: " ..
  get command
  if command = "stop" then
    exit
  elsif command = "sing" then
    put "la la la"
  elsif command = "cry" then
    put "boo hoo"
  elsif command = "laugh" then
    put "ha ha ha"
  else
    put "I don't understand"
  end if
end loop
put "That's all folks"
```

Here is a sample Execution window.

```
Enter command cry
boo hoo
Enter command laugh
```
ha ha ha
Enter command smile
I don't understand
Enter command stop
That's all folks

Notice that the response to the command `stop` is just `exit` which takes you out of the `loop` to the statement after `end loop`.

9.6 Selecting from a Menu of Commands

Sometimes it is easier to present a list or menu of all the possible commands available at a particular time and let the user choose one. If the commands are numbered you can choose by typing the number. If the command is a number we can use a `case` statement instead of an `if..then..elsif..else` statement.

Here is a program to produce the same results as before.

```plaintext
% The "ShowEmotion2" program
% Here we select commands by number from a menu
var command : int
loop
  put "Choose from 1-sing, 2-cry, 3-laugh, 4-stop: ".
  get command
  case command of
    label 1 : put "la la la"
    label 2 : put "boo hoo"
    label 3 : put "ha ha ha"
    label 4 : exit
    label : put "I don't understand"
  end case
end loop
put "That's all folks"
```

You can use strings as well as integers in a case construct. Here is a version of the `ShowEmotion2` program that asks the user to enter a string to represent the action.
% The "ShowEmotion3" program
% Here we select commands by entering a string

var command : string

loop
    put "Choose from sing, cry, laugh, stop: " ..
    get command
    case command of
        label "sing" : put "la la la"
        label "cry" : put "boo hoo"
        label "laugh" : put "ha ha ha"
        label "stop" : exit
        label : put "I don't understand"
    end case
end loop

put "That's all folks"

---

9.7 Exercises

1. Write a program to divide a class of students in two groups: those whose last names begin with A to H and those that begin with I to Z. Ask a person to enter their last name and then output a message indicating which group they are in. Repeat for each student.

2. Prepare a check for someone eating lunch in a restaurant. If the meal costs more than $4.00 a 7% tax is to be added.

3. Write a program to classify athletes into three classes by weight. The categories are: over 80 kg - heavyweight, between 60 and 80 kg - medium weight, and less than 60 kg - lightweight. Prepare the program so that a team of 10 athletes can enter their weight one after another and be told what category they are in.

4. Write a program to read a series of first names of people. After reading the series (you will need a signal), output the name that is alphabetically last.

5. Write a program to display a multiple choice question with five different answers in the window and then depending on which
answer is chosen give a different comment. For example, your question might be

Turing is:

(1) a great programming language
(2) a kind of car
(3) a mathematician
(4) a machine
(5) all of the above

The choice will be an integer from 1 to 5. Use a case construct for this.

6. Write a program to give proper greetings to a person. Ask what the occasion is and then give the appropriate greeting. You can offer a menu of available occasions. Use a case construct with string input.

7. Federal income tax is to be levied in stages on taxable income. On the first $27,500 you pay 17%, on the next $27,500 you pay 24%, and on the rest 29%. Write a program to read in a taxable income and compute the federal tax payable to the nearest cent.

8. Write a program to read in a series of positive integers and output the range of the integers, that is, the interval from the smallest to the largest.

9. Compute values of the function

\[ f(x) = 3x^2 - 2x + 1 \]

in steps of 0.1 between x=0 and x=1 inclusive and find the value of x for which f(x) is a minimum.

10. Rewrite the Grade program of this chapter to use an assert statement instead of a nested if...then construct. What happens when you enter a mark that is negative or greater than 100. Try misspelling elsif as else if and see what happens.

11. Marks in a test are given out of 10 where 9 or 10 is A grade, 7 is B grade, 6 is C grade, 5 is D grade, below 5 is F grade. Use a case construct to change the numerical test mark into the
appropriate letter grade. Note that if you want to give the same grade for both 9 and 10 you can label that case with two values. For example,

```
label 9, 10: put "This is an A"
```

Arrange to enter a number of test marks for a class ending with a mark of −1. At the end, output the percentage of the class with each letter grade. To do this you must keep track of the total numbers of marks and the numbers in each grade.

12. Write a program called *Mystery* which offers a menu of mysterious alternatives. Use a `case` construct to do quite different things in response to the different choices.

13. Write a program to input an integer from the user and output all its factors and whether or not it is a prime number. Remember that a prime number has only two factors, 1 and itself. Three is prime while four is not.

14. Write a program to output all the prime numbers from 1 to 50.

15. Write a program that asks the user for 10 marks (percents) and, at end of the run, outputs the highest mark and the lowest mark.

16. Write a program to input a series of positive integers until a sentinel is entered. Use −1 as the sentinel. Output the product of all the integers input.

17. Write a program to read in 5 marks that should be between 0 and 100 inclusive. Output an error message if a mark is not between 0 and 100. For the valid marks, output "good" if the mark is between 70 and 100 inclusive and "satisfactory" if the mark is between 50 and 69 inclusive.

### 9.8 Technical Terms

- **selection construct**
- **if...then...else construct**
- **assert statement**
- **two-way selection**
- **three-way selection**
if...then...elsif...then...else
construct

multi-way selection
case construct
label
improper data
menu