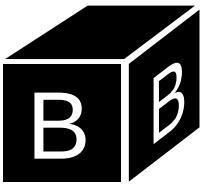


Chapter *6*

Selection

Operator	Meaning
=	Equal to
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to
#	Not equal to

Figure 6.1
The relational operators.

p	*q*	*p* & *q*
TRUE	TRUE	TRUE
TRUE	FALSE	FALSE
FALSE	TRUE	FALSE
FALSE	FALSE	FALSE

(a) The & operator.

p	*q*	*p* OR *q*
TRUE	TRUE	TRUE
TRUE	FALSE	TRUE
FALSE	TRUE	TRUE
FALSE	FALSE	FALSE

(b) The OR operator.

p	~ *p*
TRUE	FALSE
FALSE	TRUE

(c) The ~ operator.

Figure 6.2
Truth tables for the boolean operators.

Operator	Inverse Operator
=	#
<	>=
<=	>

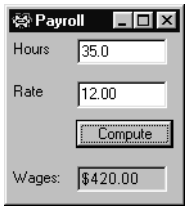
Figure 6.3
The inverses of the relational operators.

De Morgan's laws are

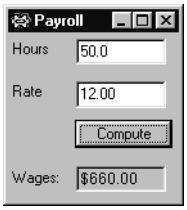
~(*p* OR *q*) = ~*p* & ~*q*
~(*p* & *q*) = ~*p* OR ~*q*

Operator	Precedence
~	Highest
&, DIV, MOD, /, *	↓
OR, +, -	
=, #, <, >, <=, >=	Lowest

Figure 6.4
Precedence of the Component Pascal operators.



(a) Without overtime.



(b) With overtime.

Figure 6.5
The dialog box for a payroll calculation.

```
MODULE Pbox06A;
    IMPORT Dialog, PboxStrings;
    VAR
        d*: RECORD
            hours*, rate*: REAL;
            message-: ARRAY 32 OF CHAR
        END;

    PROCEDURE ComputeWages*;
        VAR
            wages: REAL;
            wageString: ARRAY 32 OF CHAR;
    BEGIN
        wages := d.hours * d.rate;
        IF d.hours > 40.0 THEN
            wages := wages + (d.hours - 40.0) * 0.5 * d.rate
        END;
        PboxStrings.RealToString(wages, 1, 2, wageString);
        d.message := "$" + wageString;
        Dialog.Update(d)
    END ComputeWages;

BEGIN
    d.hours := 0.0; d.rate := 0.0;
    d.message := ""
END Pbox06A.
```

Figure 6.6
A payroll calculation program. It uses an IF statement without an ELSE part.

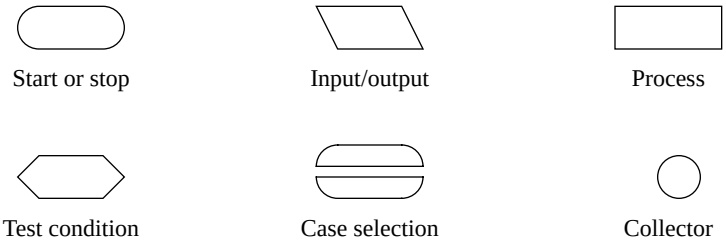


Figure 6.7
The flowchart symbols.

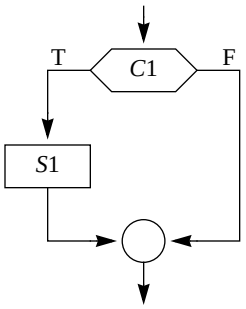


Figure 6.8
The flowchart for an IF statement without an ELSE part.

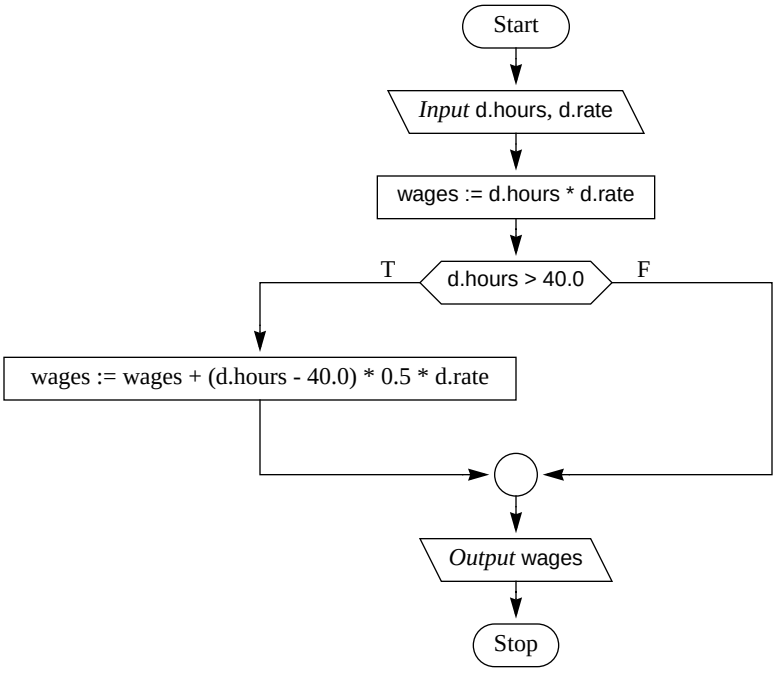


Figure 6.9
The flowchart for the program of Figure 6.6.

```
MODULE Pbox06B;
    IMPORT Dialog, PboxStrings;
    VAR
        d*: RECORD
            hours*, rate*: REAL;
            message-: ARRAY 32 OF CHAR
        END;

    PROCEDURE ComputeWages*;
        VAR
            wages: REAL;
            wageString: ARRAY 32 OF CHAR;
    BEGIN
        IF d.hours <= 40.0 THEN
            wages := d.hours * d.rate
        ELSE
            wages := 40.0 * d.rate + (d.hours - 40.0) * 1.5 * d.rate
        END;
        PboxStrings.RealToString(wages, 1, 2, wageString);
        d.message := "$" + wageString;
        Dialog.Update(d)
    END ComputeWages;

BEGIN
    d.hours := 0.0; d.rate := 0.0;
    d.message := ""
END Pbox06B.
```

Figure 6.10
A payroll calculation program that uses an IF statement with an ELSE part.

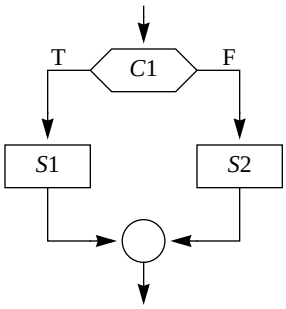
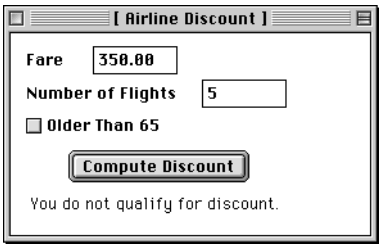


Figure 6.11
The flowchart for an IF statement with an ELSE part.

(a) Fare with discount.

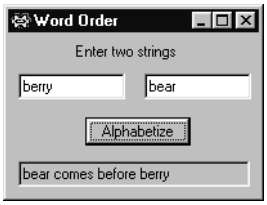


(b) Fare without discount.

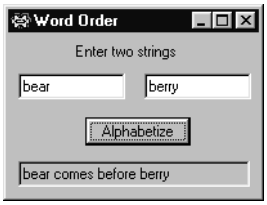
Figure 6.12
The dialog box for computing a possible discount for an airline fare.

```
MODULE Pbox06C;
    IMPORT Dialog, PboxStrings;
    VAR
        d*: RECORD
            fare*: REAL;
            numFlights*: INTEGER;
            olderThan65*: BOOLEAN;
            message-: ARRAY 64 OF CHAR
        END;

    PROCEDURE FlightDiscount*;
        CONST
            discount = 0.15;
            flightLimit = 4;
        VAR
            fare: REAL;
            fareString: ARRAY 32 OF CHAR;
    BEGIN
        IF (d.numFlights > flightLimit) & d.olderThan65 THEN
            fare := (1.0 - discount) * d.fare;
            PboxStrings.RealToString(fare, 1, 2, fareString);
            d.message := "Discounted fare: $" + fareString
        ELSE
            d.message := "You do not qualify for discount."
        END;
        Dialog.Update(d)
    END FlightDiscount;

BEGIN
    d.fare := 0.0; d.numFlights := 0;
    d.olderThan65 := FALSE;
    d.message := ""
END Pbox06C.
```

Figure 6.13
A program to compute the discount on an airline ticket.



(a) First “berry”, then “bear”.



(b) First “bear”, then “berry”.

Figure 6.14
A dialog box for comparing strings in alphabetic order.

```
MODULE Pbox06D;
    IMPORT Dialog, PboxStrings;
    VAR
        d*: RECORD
            string1*, string2*: ARRAY 16 OF CHAR;
            message-: ARRAY 64 OF CHAR;
        END;

    PROCEDURE Alphabetize*;
    BEGIN
        IF d.string1 < d.string2 THEN
            d.message := d.string1 + " comes before " + d.string2
        ELSE
            d.message := d.string2 + " comes before " + d.string1
        END;
        Dialog.Update(d)
    END Alphabetize;

BEGIN
    d.string1 := ""; d.string2 := ""
END Pbox06D.
```

Figure 6.15
A program to compare two strings.

Figure 6.16
Erroneous output from the program of Figure 6.15.

```
IF (d.numFlights > flightLimit) & (d.olderThan65 = TRUE) THEN
```

```
IF (d.numFlights > flightLimit) & d.olderThan65 THEN
```

```
IF exempt = FALSE THEN
```

```
IF ~exempt THEN
```

```
IF Condition 1 THEN
    Statement 1 ;
    Statement 2
ELSE
    Statement 3 ;
    Statement 2
END
```

```
IF Condition 1 THEN
    Statement 1
ELSE
    Statement 3
END;
Statement 2
```

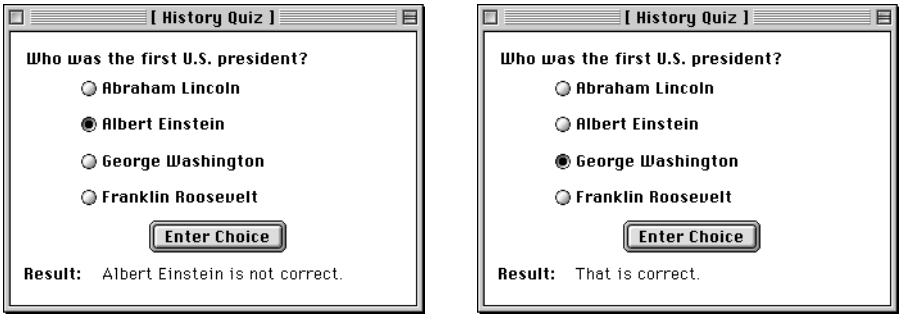
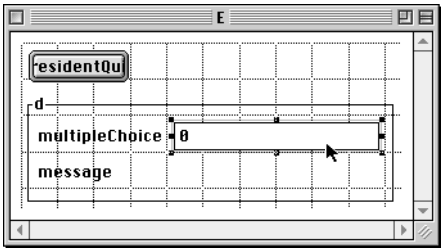


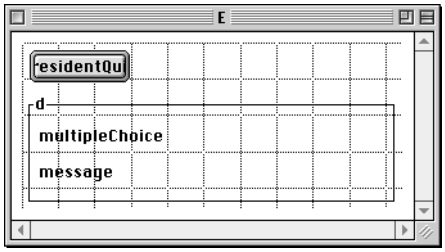
Figure 6.17
A dialog box with a set of four radio buttons.

```
MODULE Pbox06E;
    IMPORT Dialog;
    VAR
        d*: RECORD
            multipleChoice*: INTEGER;
            message-: ARRAY 64 OF CHAR
        END;

    PROCEDURE PresidentQuiz*;
    BEGIN
        CASE d.multipleChoice OF
        0:
            d.message := "Abraham Lincoln is not correct." |
        1:
            d.message := "Albert Einstein is not correct." |
        2:
            d.message := "That is correct." |
        3:
            d.message := "Franklin Roosevelt is not correct."
        END;
        Dialog.Update(d)
    END PresidentQuiz;

BEGIN
    d.multipleChoice := 0;
    d.message := ""
END Pbox06E.
```

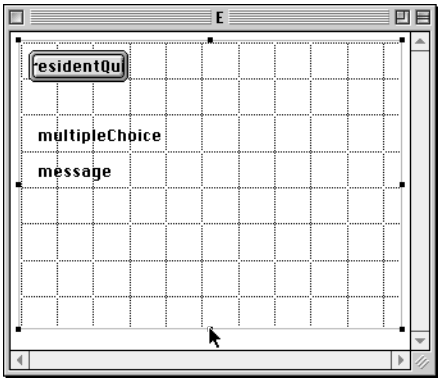
Figure 6.18
A module that takes its input from a set of radio buttons. It uses a CASE statement.



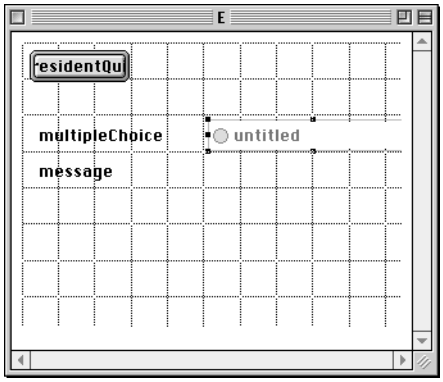
(a) Select the integer field provided by the forms generator.



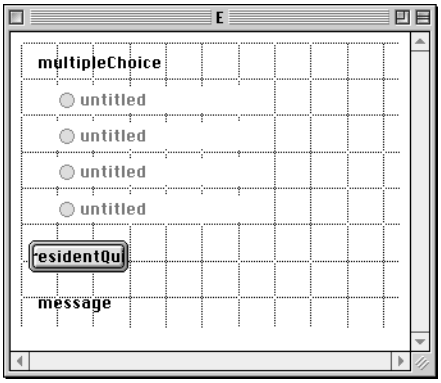
(b) Delete the integer field by pressing the delete key.



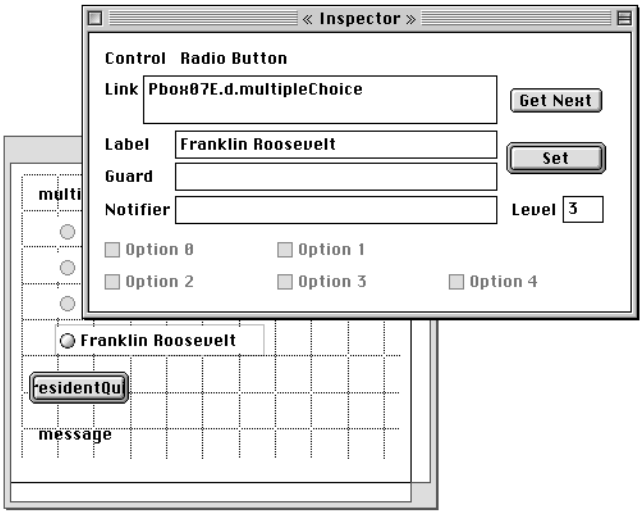
(c) Select the document and enlarge it to make room for the radio buttons.



(d) Insert the first radio button by selecting Controls→Insert Radio Button.



(e) Insert three more radio buttons and arrange them.



(f) Set the proper links manually with the component Inspector.

Figure 6.19
The process of constructing a dialog box with four radio buttons.

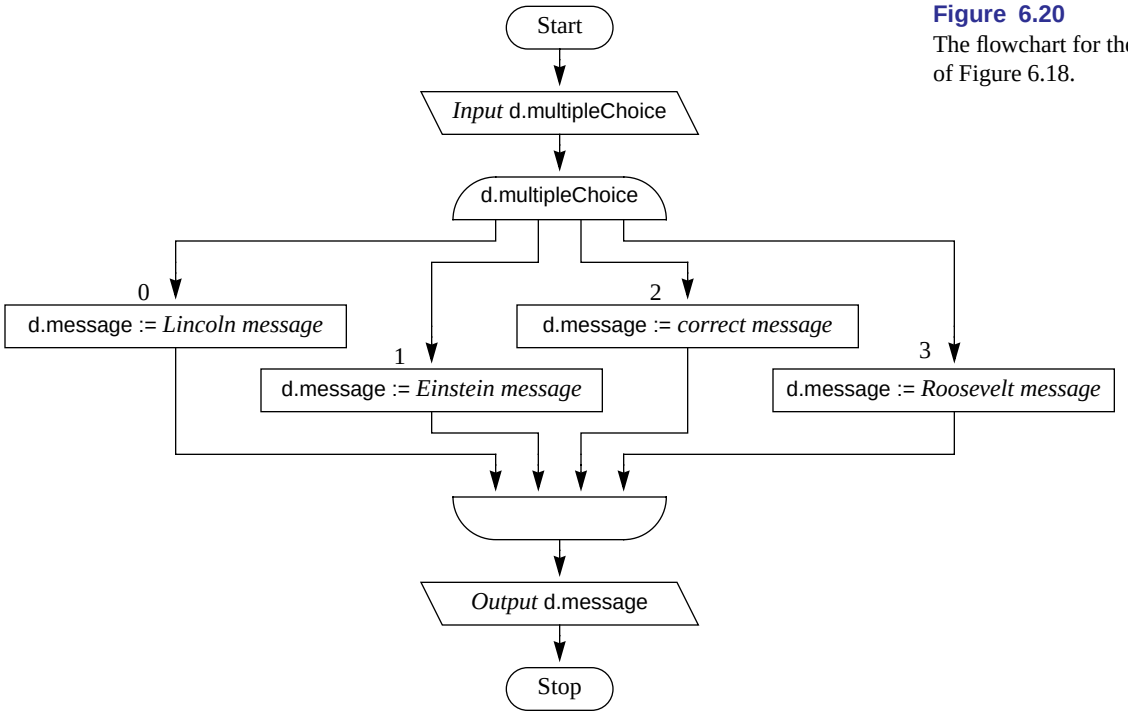


Figure 6.20
The flowchart for the module of Figure 6.18.

CP	GCL
&	∧
OR	∨
~	¬

Figure 6.21
The boolean operators in GCL.

```
IF d.hours > 40.0 THEN
    wages := wages + (d.hours - 40.0) * 0.5 * d.rate
END
```

```
if   h > 40.0 → w := w + (h – 40.0) * 0.5 * r
 []  h ≤ 40.0 → skip
fi
```

```
IF d.hours <= 40.0 THEN
    wages := d.hours * d.rate
ELSE
    wages := 40.0 * d.rate + (d.hours - 40.0) * 1.5 * d.rate
END
```

if $h \leq 40.0 \rightarrow w := d * r$
[] $h > 40.0 \rightarrow w := 40.0 * r + (h - 40.0) * 1.5 * r$
fi