Chapter 4

SELECTION STRUCTURES

Chapter 4

- Selection criteria
- The *if-else* statement
- Nested *if* statement
- The *switch* statement
- Conditional expressions

Overview

- The flow of control means the order in which a program’s statements are executed.
- Unless directed otherwise, the normal flow of control for all programs is ***sequential.***
- Selection, repetition and function invocation structures permit the flow of control to be ***altered*** in a defined way.
- In this chapter, you learn to use selection structures in C++

SELECTION CRITERIA

- **Selection criteria is the value of an expression which is used to select an appropriate flow of control**
- **In C++, there are two kinds of selection structures:**
  - If-statement: uses only 2 values, i.e. true/false or zero/non-zero
  - Switch-statement: uses multiple discrete values, i.e. integer or char or ….

Comparison Operators

Comparison operators are used to compare two operands for equality or to determine if one numeric value is greater than another.

A Boolean value of *true* or *false* is returned after two operands are compared.

C++ uses a *nonzero* value to represent a *true* and a *zero* value to represent a *false* value

Operator	**Description**	**Examples**
==	equal	a ==‘y’
!=	not equal	m!= 5
>	greater than	a*b > 7
<	less than	b < 6
<=	less than or equal	b <= a
>=	greater than or equal	c >= 6

Logical operators

- **Logical operators are used for creating more complex conditions. Like comparison operators, a Boolean value of *true* or *false* is returned after the logical operation is executed.**

Operator	Description
&&	AND
||	OR
!	NOT

- **Example:**

```
(age > 40) && (term < 10)
(age > 40) || (term < 10)
!(age > 40)
( i==j) || (a < b) || complete
```

Operator precedence

- **The relational and logical operators have a hierarchy of execution similar to the arithmetic operators.**

Level	**Operator**	**Associativity**
1.	! unary - ++ --	Right to left
2.	* / %	Left to right
3.	+ -	Left to right
4.	< <= > >=	Left to right
5.	== !=	Left to right
6.	&&	Left to right
7.	||	Left to right
8.	= += -= *= /=	Right to left

- **Example: Assume the following declarations:**

```
char key = 'm';
int i = 5, j = 7, k = 12;
double x = 22.5;
```

Expression	Equivalent	Value	Interpretation
i + 2 == k-1	(i + 2) == (k –1)	0	false
'a' +1 == 'b'	('a' +1) == 'b'	1	true
25 >= x + 1.0	25 >= (x + 1.0)	1	true
key –1 > 20	(key –1) > 20	0	false

Order of evaluation

The following compound condition is evaluated as:

```
(6*3 == 36/2) || (13<3*3 + 4) && !(6-2 < 5)
      (18 == 18) || (13 < 9 + 4) && !(4 < 5)
               1  ||       (13 < 13)  && ! 1
              1  ||   0   && 0
                1   || 0
                 1
```

The *bool* Data Type

- **As specified by the ANSI/ISO standard, C++ has a built-in Boolean data type, *bool*, containing the two values *true* and *false*.**
- **The actual values represented by the bool values, *true* and *false*, are the integer values 1 and 0, respectively.**
- **Example 4.1.1**

```
#include<iostream.h>
int main()
{
   bool t1, t2;
   t1 = true;
   t2 = true;
   cout << “The value of t1 is “<< t1
         << “\n and the value of t2 is “<< t2 << endl;
   return 0;
}
```

THE if-else STATEMENT

The *if-else* statement directs the computer to select a statement based on the result of a comparison.

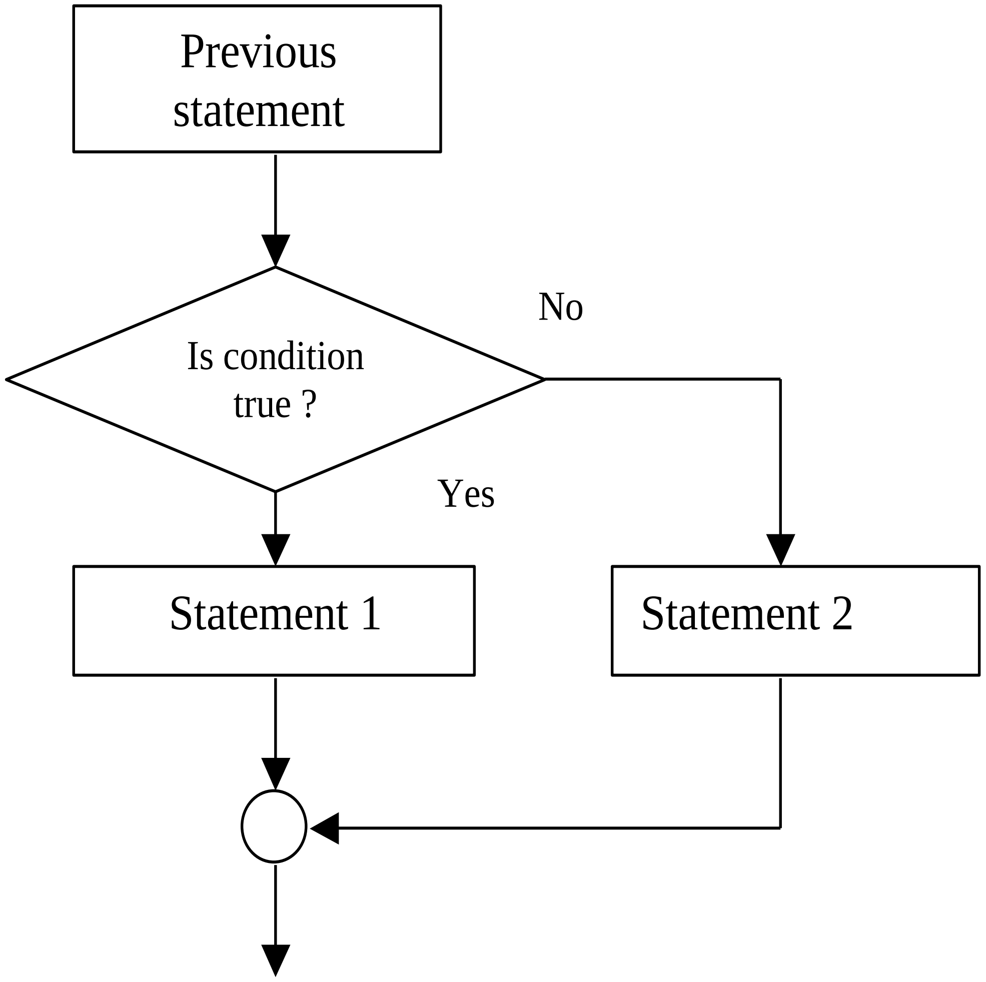
The syntax:

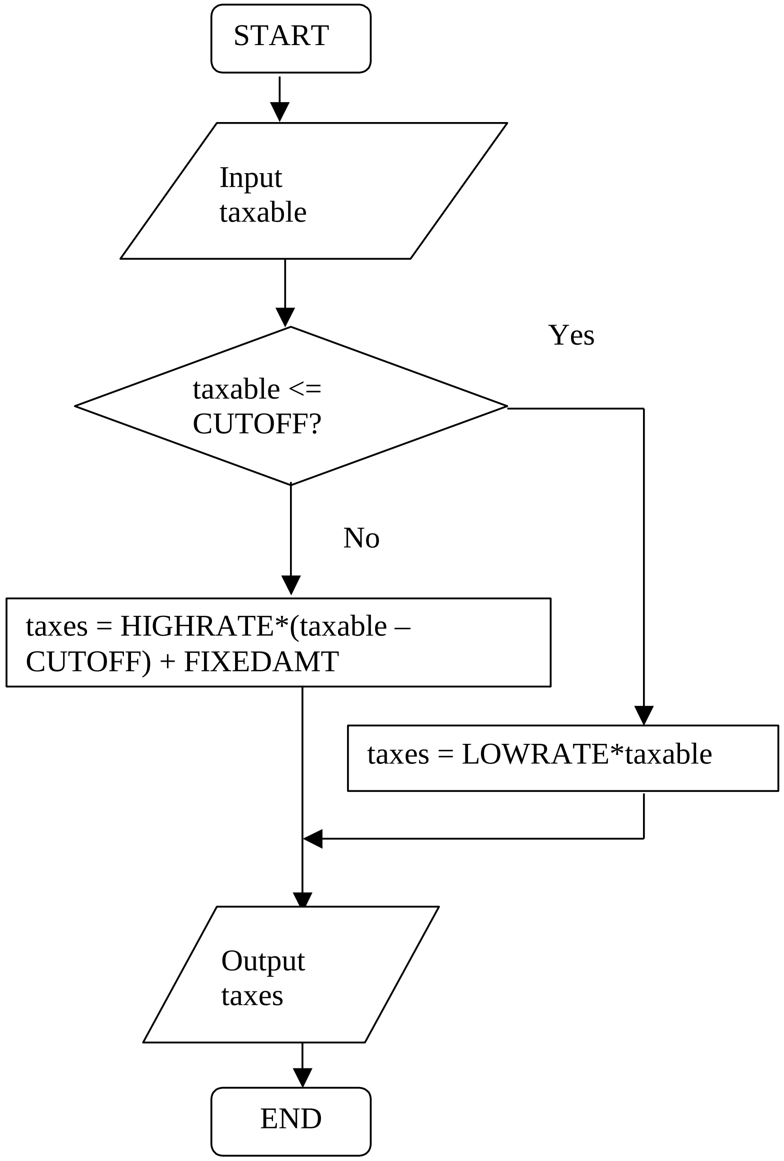
if *(conditional expression)*

  statement 1

else

  statement 2





Example 4.2.1

We construct a C++ program for determining income taxes.

Assume that these taxes are assessed at 2% of taxable incomes less than or equal to $20,000. For taxable income greater than $20,000, taxes are 2.5% of the income that exceeds $20,000 plus a fixed amount of $400.

Example 4.2.1

```
#include <iostream.h>
#include <iomanip.h>
const float LOWRATE = 0.02;     // lower tax rate
const float HIGHRATE = 0.025;   // higher tax rate
const float CUTOFF = 20000.0;   // cut off for low rate
const float FIXEDAMT = 400;
int main()
{
  float taxable, taxes;
  cout << "Please type in the taxable income: ";
  cin  >> taxable;
   if (taxable <= CUTOFF)
    taxes = LOWRATE * taxable;
  else
    taxes = HIGHRATE * (taxable - CUTOFF) + FIXEDAMT;
// set output format
  cout << setiosflags(ios::fixed)
       << setiosflags(ios::showpoint)
       << setprecision(2);
  cout << "Taxes are $ " << taxes << endl;
 return 0;
}
```

The results of the above program:

```
Please type in the taxable income: 10000
Taxes are $ 200
```

and

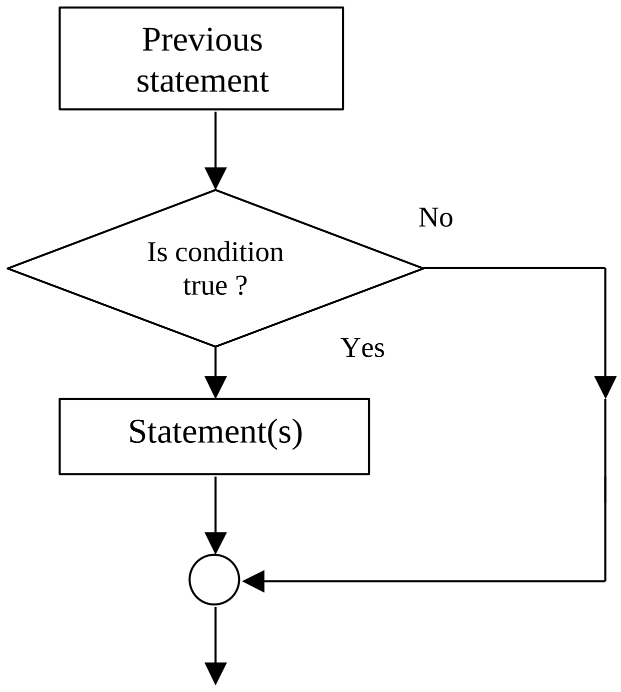
```
Please type in the taxable income: 30000
Taxes are $ 650
```

One-way Selection

- A useful modification of the *if-else* statement involves omitting the *else* part of the statement. In this case, the *if* statement takes a shortened format:

  if (conditional expression)

  statement;



Example 4.2.2

The following program displays an error message for the grades that is less than 0 or more than 100.

```
#include <iostream.h>
int main()
{
   int grade;

  cout << "\nPlease enter a grade: ";
  cin  >> grade;

  if(grade < 0 || grade > 100)
    cout << " The grade is not valid\n";
  return 0;
}
```

Exercise

NESTED if STATEMENT

- The inclusion of one or more *if* statement within an existing if statement is called a *nested if statement*.
- The if-else Chain

  When an *if* statement is included in the *else* part of an existing *if* statement, we have an *if-else chain*.

```
if (Expression 1)
   Statement 1
else if (Expression 2)
   Statement 2
else
   Statement 3
```

Example 4.3.1

```
// This program can solve quadratic equation
#include <iostream.h>
#include <math.h>
#include <iomanip.h>
int main()
{
   double a, b, c, del, x1, x2;
    cout << “Enter the coefficients of the equation: “<< endl;
    cin  >> a >> b >> c;
    del = b*b – 4.0*a*c;
    if (del == 0.0)
    {
        x1 = x2 = -b/(2*a);
        cout << "x1 = “ << x1 << setw(20) << “x2 = “ << x2 << endl;
     }
```

```
 else if (del > 0.0)
    {
        x1 = (-b + sqrt(del))/(2*a);
        x2 = (-b – sqrt(del))/(2*a);
        cout << "x1 = “ << x1 << setw(20) << “x2 = “ << x2 << endl;
     }
    else
    cout << "There is no solution\n";
   return 0;
}
```

The output of the above program:

```
Enter the coefficients of the equation:
1    5   6
x1 =  -2.0          x2  = -3.0
```

NESTED if STATEMENT (cont'd)

- The dangling **else**

An *else* part which can ambiguously attach to any in 2 *if* statements is called **the dangling else**

if (exp1) **if** (exp2) statement1 **else** statement2

- To solve the problem of dangling else, you can use:
  - C++ convention: *else* part is attached to the nearby *if* statement
  - Compound statement:

if (exp1) { **if** (exp2) statement1 } **else** statement2

Or

if (exp1) {**if** (exp2) statement1 **else** statement2 }

Exercise

THE switch STATEMENT

- **The *switch* statement controls program flow by executing a set of statements depending on the value of an expression.**

- **The syntax for the *switch* statement:**

```
switch(expression){
   case label1:
         statement(s) 1;
         break;
  case label2;
         statement(s) 2;
         break;
  default:
         statement(s) 3;
}
```

<u>Note</u>: The value of expression must be an integer data type, which includes the *char, int*, *long int*, and *short* data types.

Execution of the *switch* statement

- The expression in the *switch* statement must evaluate to an integer result.

- The *switch* expression’s value is compared to each of these case values in the order in which these values are listed until a match is found. When a match occurs, execution begins with the statement following the match.

- If the value of the expression does not match any of the case values, no statement is executed unless the keyword *default* is encountered. If the value of the expression does not match any of the case values, program execution begins with the statement following the word *default*.

break statements in the *switch* statement

- The *break* statement is used to identify the end of a particular case and causes an immediate exit from the *switch* statement.

- If the *break* statements are omitted, all cases following the matching case value, including the default case, are executed.

<u>Example 4.4.1</u>

```
#include <iostream.h>
int main()
{
  int iCity;

  cout << "Enter a number to find the state where a city is located. "<<
     endl;
  cout << “1. Boston” << endl;
  cout << "2. Chicago" << endl;
  cout << "3. Los Angeles” << endl;
  cout << "4. Miami” << endl;
  cout << "5. Providence” << endl;
  cin  >> iCity;
  switch (iCity)
  {
    case 1:
      cout << "Boston is in Massachusetts " << endl;
      break;
    case 2:
      cout << "Chicago is in Illinois " << endl;
      break;
    case 3:
     cout << "Los Angeles is in California " << endl;
     break;
    case 4:
     cout << "Miami is in Florida " << endl;
     break;
    case 5:
     cout << "Providence is in Rhode Island " << endl;
     break;
    default:
     cout << “You didn’t select one of the five cities” << endl;
  }    // end of switch
  return 0;
}
```

The output of the above program:

```
Enter a number to find the state where a city is located.
1. Boston
2. Chicago
3. Los Angeles
4. Miami
5. Providence

3
Los Angeles is in California
```

When writing a *switch* statement, you can use multiple case values to refer to the same set of statements; the *default* label is optional.

```
switch(number)
{
   case 1:
      cout << “Have a Good Morning\n”;
      break;
   case 2:
      cout << “Have a Happy Day\n”;
      break;
   case 3:
   case 4:
   case 5:
      cout << “Have a Nice Evening\n”;
 }
```

CONDITIONAL EXPRESSIONS

- **A conditional expression uses the conditional operator, ?:, and provides an alternative way of expressing a simple *if-else* statement.**

- **The syntax of a conditional expression:**

  expression1 ? expression2 : expression3

- **If the value of *expression1* is nonzero (true), *expresson2* is evaluated; otherwise, *expression3* is evaluated. The value for the complete conditional expression is the value of either *expression2* or *expression3* depending on which expression was evaluated.**

- **Example: The *if* statement:**

```
if (hours > 40)
    rate = 0.45;
else
     rate = 0.02;
```

can be replaced with the following one-line statement:

```
rate = (hours > 40) ? 0.45 : 0.02;
```

THE enum SPECIFIER

- **The *enum* specifier creates an enumerated data type, which is simply a user-defined list of values that is given its own data type name.**

- **Such data types are identified by the reserved word *enum* followed by an optional user-selected name for the data type and a listing of acceptable values for the data type.**

- **Example:**

```
enum day { mon, tue, wed, thr, fri, sat, sun}
enum color {red, green, yellow};
```

- **Any variable declared to be of type *color* can take only a value of *red* or *green* or *yellow*. Any variable declared to be of type *day* can take only a value among seven given values.**

  The statement

  ```
  enum day a, b,c;
  ```

  declares the variables *a, b*, and *c* to be of type *day*.

- **Internally, the acceptable values of each enumerated data type are ordered and assigned sequential integer values beginning with 0.**

- **<u>Example</u>: For the values of the type *color*, the correspondences created by C++ compiler are that *red* is equivalent to 0, *green* is equivalent to 1, and *yellow* is equivalent to 2.**

- **The equivalent numbers are required when inputting values or displaying values.**

Example 4.6.1

```
#include <iostream.h>
int main(){
   enum color{red, green, yellow};
   enum color crayon = red;
   cout << “\nThe color is “ << crayon << endl;
   cout << “Enter a value:  “;    cin >> crayon;
   if (crayon == red)
      cout << “The crayon is  red.” << endl;
   else if (crayon == green)
      cout << “The crayon is green.” << endl;
   else if (crayon== yellow)
      cout << “The crayon is yellow.” << endl;
   else
      cout << “The color is not defined. \n” <<
     endl;
   return 0;
}
```

The output of the above program:

```
The color is 0
Enter a value: 2
The crayon is yellow.
```

Exercise

Summary

- **Selection criteria is used to select a flow of control**
- **In C++, there are**
  - **if** statetement (2 choices)
  - **if-else** statement (2 choices)
  - **switch** statement (many choices)
- **Some other concepts:**
  - Conditional expression
  - **enum** type