Chapter 4: Making Decisions

Starting Out with C++
Early Objects
Seventh Edition

by Tony Gaddis, Judy Walters,
and Godfrey Muganda

Topics

Topics (continued)

4.1 Relational Operators

- Used to compare numbers to determine relative order
- Operators:

`>`	Greater than
`<`	Less than
`>=`	Greater than or equal to
`<=`	Less than or equal to
`==`	Equal to
`!=`	Not equal to

Relational Expressions

- Relational expressions are Boolean (evaluate to **`true`** or **`false`**)
- Examples:

  `12 > 5` is **`true`**

  `7 <= 5` is **`false`**

The **==** operator determines whether the operand on its left is equal to the operand on its right.

If both operands have the same value, the expression is true.

if **x** is 10, then

x == 10 is **true**

x != 8 is **true**

x == 8 is **false**

Example Relational Expressions

(Assume x is 10 and y is 7.)

Expression	Value
x < y	
x > y	
x >= y	
x <= y	
y != x	

Relational Expressions

- Can be assigned to a Boolean variable

```
bool result = (x <= y);
```

- Assigns `0` for **`false`**, `1` for **`true`**
- Do not confuse **`=`** (assignment) and

`==` (equal to)

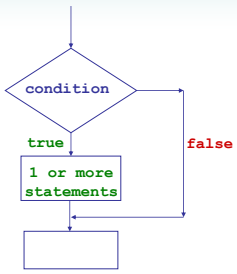
```
bool areEqual = (x == y);
```


4.2 The `if` Statement

```
if (condition)
 {
   statement1;
   statement2;
      …
   statementn;
 }
```

- If **`(condition)`** is **`true`**, then the ***`statement(s)`*** in the body are executed.
- If **`(condition)`** is **`false`**, then the ***`statement(s)`*** are skipped.

`if` Statement Flow of Control



Format of the `if` Statement

```
if (condition)
 {
   statement1;
   statement2;
      …
   statementn;
 }
```

No ; goes here

; goes here

The block inside the braces is called the body of the **`if`** statement.

If there is only 1 statement in the body, the **`{ }`** may be omitted.

Example `if` Statements

```
if (score >= 65)
   cout << "You passed.\n";

if (score >= 90)
{
   grade = 'A';
   cout << "Wonderful job!\n";
}
```


Program 4-2

```
// This program correctly averages 3 test scores.
#include <iostream>
#include <iomanip>
using namespace std;

int main()
{
   int score1, score2, score3;
   double average;

   // Get the three test scores
   cout << "Enter 3 test scores and I will average them: ";
   cin >> score1 >> score2 >> score3;

   // Calculate and display the average score
   average = (score1 + score2 + score3) / 3.0;
   cout << fixed << showpoint << setprecision(1);
   cout << "Your average is " << average << endl;

   // If the average equals
   if (average == 100)
   {
      cout << "Congratulati
      cout << "That's a perf
   }
   return 0;
}
```

```
Enter 3 test scores and I will average them: 80 90 70[Enter]
Your average is 80.0

Enter 3 test scores and I will average them: 100 100 100[Enter]
Your average is 100.0
Congratulations! That's a perfect score!
```

```
if (average == 100)
{
   cout << "Congratulations! ";
   cout << "That's a perfect score!\n";
}
```

There are four important things to notice.

First, the word if, which begins the statement, is a C++ key word and must be written in lowercase.

Second, the condition to be tested (average == 100) must be enclosed inside parentheses.

Third, there is no semi-colon after the test condition, There is a semi-colon after each action associated with the if construct..

And finally, the block of statements to be conditionally executed is surrounded by curly braces. This is required whenever two or more actions are associated with an if statement.

```
if (hours > 40)
{
   overTime = true;
   PayRate *= 2;
}
```

```
if (temperature > 32)
   freezing = false;
```

```
if (condition)
{
   statement1;
   statement2;
   …
   statementn;
}
```

No ; goes here

; goes here

Three common errors you must watch out for.

1. Misplaced semicolons
2. Missing braces
3. Confusing = with ==

What is `true` and `false`?

- An expression whose value is 0 is considered **`false`**.
- An expression whose value is non-zero is considered **`true`**.
- An expression need not be a comparison – it can be a single variable or a mathematical expression.

Flag

- A variable that signals a condition
- Usually implemented as a **`bool`**
- Meaning:
  - **`true`**: the condition exists
  - **`false`**: the condition does not exist
- The flag value can be both set and tested with **`if`** statements

Flag Example

Example:

```
bool validMonths = true;
     …
if (months < 0)
   validMonths = false;
   …
if (validMonths)
   moPayment = total / months;
```


Comparisons with floating-point numbers

- It is difficult to test for equality when working with floating point numbers.
- It is better to use
  - greater than, less than tests, or
  - test to see if value is very close to a given value

4.3 The `if/else` Statement

- Allows a choice between statements depending on whether `(condition)` is `true` or `false`
- Format:

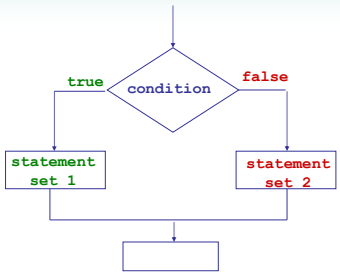
```
if (condition)
{
   statement set 1;
}
else
{
   statement set 2;
}
```


How the `if/else` Works

```
if (condition)
{
   statement set 1;
}
else
{
   statement set 2;
}
```

- If `(condition)` is `true`, *`statement set 1`* is executed and *`statement set 2`* is skipped.
- If `(condition)` is `false`, *`statement set 1`* is skipped and *`statement set 2`* is executed.

`if/else` Flow of Control



Example `if/else` Statements

```
if (score >= 60)
  cout << "You passed.\n";
else
  cout << "You did not pass.\n";


if (intRate > 0)
{  interest = loanAmt * intRate;
   cout << interest;
}
else
  cout << "You owe no interest.\n";
```


Program 4-5

```
// This program uses the modulus operator to determine
// if a number is odd or even. If the number is evenly divisible
// by 2, it is an even number. A remainder indicates it is odd.
#include <iostream>
using namespace std;
int main()
{
   int number;
   cout << "Enter an integer and I will tell you if it\n";
   cout << "is odd or even. ";
   cin >> number;

   if (number % 2 == 0)
      cout << number << " is even.\n";
   else
      cout << number << " is odd.\n";
   return 0;
}
```

```
Enter an integer and I will tell you if it
is odd or even. 17[Enter]
17 is odd.
```

Program 4-6

```
// This program makes sure that the divisor is not
// equal to 0 before it performs a divide operation.
#include <iostream>
using namespace std;

int main()
{
   double num1, num2, quotient;

   // Get the two numbers
   cout << "Enter two numbers: ";
   cin >> num1 >> num2;

   // If num2 is not zero, perform the division.
   if (num2 != 0)
   {
      quotient = num1 / num2;
      cout << "The quotient of " << num1 << " divided by "
           << num2 << " is " << quotient << ".\n";
   }
   else
   {
      cout << "Division by zero is not possible.\n"
           << "Please run the program again and enter "
           << "a number other than zero.\n";
   }
   return 0;
}
```

No;

```
Enter two numbers: 10 0[Enter]
Division by zero is not possible.
Please run the program again and enter a number other than zero.
```

4.4 The `if/else if` Statement

- Chain of `if` statements that test in order until one is found to be true
- Also models thought processes

  "If it is raining, take an umbrella,
  else, if it is windy, take a hat,
  else, if it is sunny, take sunglasses."

`if/else if` Format

```
if (condition 1)
{
  statement set 1;
}
else if (condition 2)
{
  statement set 2;
}
...
else if (condition n)
{
  statement set n;
}
```


Copyright © 2011 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

```
if (condition 1)
{
  statement set 1;
}
else if (condition 2)
{
  statement set 2;
}
…
else if (condition n)
{
  statement set n;
}
```



```
if (condition 1)
{
  statement set 1;
}
else if (condition 2)
{
  statement set 2;
}
…
else if (condition n)
{
  statement set n;
}
```

Copyright © 2011 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Program 4-7

```
// This program uses an if/else if statement to assign a
// letter grade (A, B, C, D, or F) to a numeric test score.
#include <iostream>
using namespace std;

int main()
{
    int testScore;          // Holds a numeric test score
    char grade;   // Holds a letter grade

    // Get the numeric score
    cout << "Enter your numeric test score and I will\n";
    cout << "tell you the letter grade you earned: ";
    cin >> testScore;

    // Determine the letter grade
    if (testScore < 60)
        grade = 'F';
    else if (testScore < 70)
        grade = 'D';
    else if (testScore < 80)
        grade = 'C';
    else if (testScore < 90)
        grade = 'B';
    else if (testScore <= 100)
        grade = 'A';

    // Display the letter grade
    cout << "Your grade is " << grade << ".\n";
    return 0;
}
```

```
Enter your numeric test score and I will
tell you the letter grade you earned: 88[Enter]
Your grade is B.
```

Program 4-8

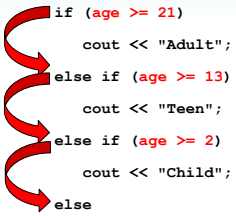
```
// This program illustrates a bug that occurs when independent
// if statements are used to assign a letter grade to a numeric test
// score.
#include <iostream>
using namespace std;

int main()
{
    int testScore; // Holds a numeric test score
    char grade; // Holds a letter grade

    // Get the numeric score
    cout << "Enter your test score and I will tell you\n";
    cout << "the letter grade you earned: ";
    cin >> testScore;

    // Determine the letter grade. What grade will be assigned?
    if (testScore < 60)
        grade = 'F';
    if (testScore < 70)
        grade = 'D';
    if (testScore < 80)
        grade = 'C';
    if (testScore < 90)
        grade = 'B';
    if (testScore <= 100)
        grade = 'A';

    // Display the letter grade
    cout << "Your grade is " << grade << ".\n";
    return 0;
}
```

```
Enter your numeric test score and I will tell you
the letter grade you earned: 40[Enter]
Your grade is A.
```

Using a Trailing `else`

- Used with `if/else if` statement when all of the conditions are false
- Provides a default statement or action
- Can be used to catch invalid values or handle other exceptional situations

Example `if/else if` with Trailing `else`



```
if (age >= 21)
    cout << "Adult";
else if (age >= 13)
    cout << "Teen";
else if (age >= 2)
    cout << "Child";
else
    cout << "Baby";
```



```
// This program uses an if/else if statement to assign a
// letter grade (A, B, C, D, or F) to a numeric test score.
// A trailing else has been added to catch test scores > 100.
#include <iostream>
using namespace std;

int main()
{
    int testScore; // Holds a numeric test score
    char grade; // Holds a letter grade
    bool goodScore = true; // Is the score valid?

    // Get the numeric score
    cout << "Enter your numeric test score and I will\n";
    cout << "tell you the letter grade you earned: ";
    cin >> testScore;

    // Determine the letter grade
    if (testScore < 60)
        grade = 'F';
    else if (testScore < 70)
        grade = 'D';
    else if (testScore < 80)
        grade = 'C';
    else if (testScore < 90)
        grade = 'B';
    else if (testScore <= 100)
        grade = 'A';
    else
        goodScore = false; // The score was greater than 100

    // If the score is valid, display the corresponding letter grade;
    // otherwise, display an error message
    if (goodScore)
        cout << "Your grade is " << grade << endl;
    else
        cout << "We do no
    return 0;
}
```

```
Enter your numeric test score and I will
tell you the letter grade you earned: 104 [Enter]
We do not give scores higher than 100.
```

4.5 Menu-Driven Program

- Menu: list of choices presented to the user on the computer screen
- Menu-driven program: program execution controlled by user selecting from a list of actions
- Menu can be implemented using `if/else if` statements

Menu-driven Program Organization

- Display list of numbered or lettered choices for actions.
- Input user's selection of number or letter
- Test user selection in `(condition)`
  - if a match, then execute code to carry out desired action
  - if not, then test with next `(condition)`

Program 4-10

```
// This menu-driven program uses an if/else statement to carry
// out the correct set of actions based on the user's menu choice.
#include <iostream>
#include <iomanip>
using namespace std;

int main()
{
   // Constants for membership rates
   const double ADULT_RATE  = 40.0;
   const double CHILD_RATE  = 20.0;
   const double SENIOR_RATE = 30.0;

   int choice;          // Menu choice
   int months;          // Number of months
   double charges;      // Monthly charges

   // Display the menu and get the user's choice
   cout << "   Health Club Membership Menu\n\n";
   cout << "1. Standard Adult Membership\n";
   cout << "2. Child Membership\n";
   cout << "3. Senior Citizen Membership\n";
   cout << "4. Quit the Program\n\n";
```

```
   cout << "Enter your choice: ";
   cin  >> choice;

   // Set the numeric output formatting
   cout << fixed << showpoint << setprecision(2);

   // Use the menu selection to execute the correct set of actions
   if (choice == 1)
   {  cout << "\nFor how many months? ";
      cin  >> months;
      charges = months * ADULT_RATE;
      cout << "The total charges are $" << charges << endl;
   }
   else if (choice == 2)
   {  cout << "\nFor how many months? ";
      cin  >> months;
      charges = months * CHILD_RATE;
      cout << "The total charges are $" << charges << endl;
   }
   else if (ch
   {  cout <<
      cin  >>
      charges
      cout <<
   }
   else if (ch
   {  cout <<
            <<
   }
   return 0;
}
```

```
   Health Club Membership Menu

1. Standard Adult Membership
2. Child Membership
3. Senior Citizen Membership
4. Quit the Program

Enter your choice: 3[Enter]
For how many months? 6[Enter]
The total charges are $180.00
```

4.6 Nested `if` Statements

- An **`if`** statement that is part of the **`if`** or **`else`** part of another **`if`** statement
- Can be used to evaluate > 1 data item or condition

```
if (score < 100)
{
   if (score > 90)
      grade = 'A';
}
```


Notes on Coding Nested `if`s

- An **`else`** matches the nearest **`if`** that does not have an **`else`**

```
if (score < 100)
   if (score > 90)
      grade = 'A';
else ...  // goes with second if,
          // not first one
```

- Proper indentation aids comprehension

Program 4-11

```
// This program determines whether a loan applicant qualifies for
// a special loan interest rate. It uses nested if/else statements.
#include <iostream>
using namespace std;

int main()
{
   char employed,       // Currently employed? (Y or N)
        recentGrad;     // Recent college graduate? (Y or N)

   // Is the applicant employed and a recent college graduate?
   cout << "Answer the following questions\n";
   cout << "with either Y for Yes or N for No.\n";

   cout << "Are you employed? ";
   cin  >> employed;
   cout << "Have you graduated from college in the past two years? ";
   cin  >> recentGrad;

   // Determine the applicant's loan qualifications
```

```
   // Determine the applicant's loan qualifications
   if (employed == 'Y')
   {
      if (recentGrad == 'Y')          // Employed and a recent grad
      {
         cout << "You qualify for the special interest rate.\n";
      }
      else                            // Employed but not a recent grad
      {
         cout << "You must have graduated from college in the past\n";
         cout << "two years to qualify for the special interest rate.\n";
      }
   }
   else                               // Not employed
   {
      cout << "You must be employed to qualify for the "
           << "special interest rate. \n";
   }
   return 0;
}
```

Program Output with Example Input Shown in Bold

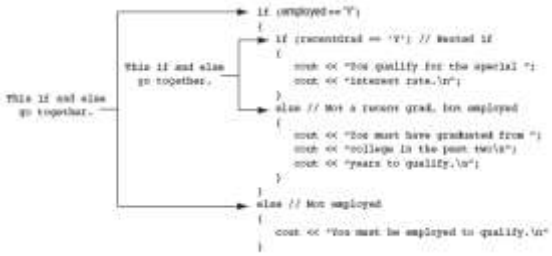
```
Answer the following questions
with either Y for Yes or N for No.
Are you employed? N[Enter]
Have you graduated from college in the past two years? Y[Enter]
You must be employed to qualify for the special interest rate.
```

Program Output with Other Example Input Shown in Bold

```
Answer the following questions
with either Y for Yes or N for No.
Are you employed? Y[Enter]
Have you graduated from college in the past two years? N[Enter]
You must have graduated from college in the past
two years to qualify for the special interest rate.
```

Program Output with Other Example Input Shown in Bold

```
Answer the following questions
with either Y for Yes or N for No.
Are you employed? Y[Enter]
Have you graduated from college in the past two years? Y[Enter]
You qualify for the special interest rate.
```



4. Areas of Rectangles

The area of a rectangle is the rectangle's length times its width. Write a program that asks for the length and width of two rectangles. The program should tell the user which rectangle has the greater area, or if the areas are the same.

5. Book Club Points

An online book club awards points to its customers based on the number of books purchased each month. Points are awarded as follows:

Books Purchased	Points Earned
0	0
1	5
2	15
3	30
4 or more	60

Write a program that asks the user to enter the number of books purchased this month and then displays the number of points awarded.

9. Software Sales

A software company sells a package that retails for $99. Quantity discounts are given according to the following table:

Quantity	Discount
10–19	20%
20–49	30%
50–99	40%
100 or more	50%

Write a program that asks for the number of units purchased and computes the total cost of the purchase.

Input Validation: Make sure the number of units is greater than 0.

10. Bank Charges

A bank charges $10 per month plus the following check fees for a commercial checking account:

$.10 each for fewer than 20 checks
$.08 each for 20–39 checks
$.06 each for 40–59 checks
$.04 each for 60 or more checks

Write a program that asks for the number of checks written during the past month, then computes and displays the bank's fees for the month.

Input Validation: Do not accept a negative value for the number of checks written.

Enter a whole number from 1 to 99.
I will find a combination of coins;
that equals that amount of change.
93

93 cents equals
3 quarters
1 dimes
1 nickels and
3 pennies

I accept only one one-dollar bill.
Enter the cost of an item (increments of 5 cents only 5 – 95).
I will find a combination of coins;
that equals the change you will get back.
85

You gave me a dollar
The item costs 85 cents
Your change is
0 quarters
1 dimes and
1 nickel

Random Numbers

Some programs need to use randomly generated numbers. The C++ library has a function called **rand()** for this purpose. To use the rand() function, you must include the **cstdlib** header file in your program. The number returned by the function is an int between 0 and the largest value int can hold.

Here is an example of how it is used.

```
int randomNum = rand();
```

Program 3-31

```
// This program demonstrates what happens in C++ if you
// try to generate random numbers without setting a "seed".
#include <iostream>
#include <cstdlib>
using namespace std;

int main ()
{
   int num1, num2, num3;  // These hold the 3 random numbers

   // Now generate and print three random numbers
   num1 = rand();
   num2 = rand();
   num3 = rand();
   cout << num1 << "      " << num2 << "      " << num3 << endl;
   return 0;
}
```

Program Output from Run 1

```
41      16467      6334
```

Program Output from Run 2

```
41      16467      6334
```

Random Numbers

However, the numbers returned by the function are really *pseudorandom*. This means they have the appearance and properties of random numbers, but in reality are not random.

They are actually generated with an algorithm. The algorithm needs a starting value, called a ***seed***, to generate the numbers. If it is not given one, it will produce the same stream of numbers each time it is run. Program 3-31 ⟶ illustrates this.

Random Numbers

- These require **cstdlib** header file
- **rand**
  - Returns a random number between 0 and the largest **int** the computer holds
  - Will yield same sequence of numbers each time the program is run
- **srand(x)**
  - Initializes random number generator with

    unsigned int x
  - Should be called at most once in a program

Program 3-32

```
// This program demonstrates random numbers, providing
// a "seed" for the random number generator.
#include <iostream>
#include <cstdlib>
using namespace std;

int main()
{
   int num1, num2, num3;  // These hold the 3 random numbers
   unsigned seed;         // Random generator seed

   // Get a "seed" value from the user
   cout << "Enter a seed value: ";
   cin  >> seed;

   // Set the random generator seed before calling rand()
   srand(seed);

   // Now generate and print three random numbers
   num1 = rand();
   num2 = rand();
   num3 = rand();
   cout << num1 << "      " << num2 << "      " << num3 << endl;
   return 0;
}
```

Program Output with Example Input Shown in Bold

```
Run 1:                                  Run 2:
Enter a seed value: 99[Enter]           Enter a seed value: 175[Enter]
130      15131      259                 567      18469      28887
```

Another common practice for getting a seed value is to call the **time**(0) function, which is part of the standard library. This function returns the number of seconds that have elapsed since midnight, January 1, 1970.

To use the time function in a program you must include the **ctime** header file, and you must pass 0 as an argument to the function when you call it.

The following code segment illustrates how to "seed" the random number generator with a value obtained this way.

```
unsigned seed;
seed = time(0);
srand(seed);
```

Or

```
srand(time(0));
```

If you wish to limit the range of the random number to an integer between 1 and maxRange, use the following formula.

```
randomNum = 1 + rand() % maxRange;
```

For example, if you wish to generate a random number in the range of 1 through 6 to represent the roll of a dice, you would use

```
dice = 1 + rand() % 6;
```

NOTE: The mod operation % gives us the remainder of an integer divide. When the integer returned by rand() is divided by 6, the remainder will be a number between 0 and 5. Because we want a number between 1 and 6, we simply add 1 to the result.

So to pull a number between 1 and 10

```
#include<cstdlib>

    srand=(time(0));

    int maxRange=10;
    randomNum = 1 + rand() % maxRange;
```

8. Math Tutor

Write a program that can be used as a math tutor for a young student. The program should display two random numbers between 10 and 50 that are to be added, such as:

```
  24
+ 12
____
```

The program should then wait for the student to enter the answer. If the answer is correct, a message of congratulations should be printed. If the answer is incorrect, a message should be printed showing the correct answer.

The way the two numbers are displayed should match the above example.

4.7 Logical Operators

Used to create relational expressions from other relational expressions

Operators, Meaning, and Explanation

`&&`	AND	New relational expression is true if both expressions are true
`||`	OR	New relational expression is true if either expression is true
`!`	NOT	Reverses the value of an expression; true expression becomes false, false expression becomes true

Logical Operator Examples

```
int x = 12, y = 5, z = -4;
```

`(x > y) && (y > z)`	
`(x > y) && (z > y)`	
`(x <= z) || (y == z)`	
`(x <= z) || (y != z)`	
`!(x >= z)`	

Logical Precedence

Highest `!`
`&&`
Lowest `||`

Example:

```
(2 < 3) || (5 > 6) && (7 > 8)
```


Checking Numeric Ranges with Logical Operators

- Used to test if a value is within a range

```
if (grade >= 0 && grade <= 100)
   cout << "Valid grade";
```

- Can also test if a value lies outside a range

```
if (grade <= 0 || grade >= 100)
   cout << "Invalid grade";
```

- Cannot use mathematical notation

```
if (0 <= grade <= 100) //Doesn't
                       //work!
```


4.9 More About Variable Definitions and Scope

- Scope of a variable is the block in which it is defined, from the point of definition to the end of the block
- Variables are usually defined at beginning of function
- They may instead be defined close to first use

More About Variable Definitions and Scope

- Variables defined inside **{ }** have local or block scope
- When in a block that is nested inside another block, you can define variables with the same name as in the outer block.
  - When the program is executing in the inner block, the outer definition is not available
  - This is generally not a good idea

4.10 Comparing Characters and Strings

- Can use relational operators with characters and string objects

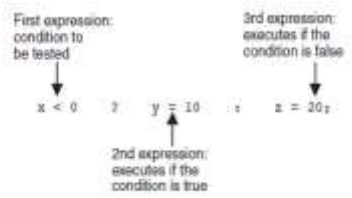
```
if (menuChoice == 'A')
if (firstName == "Beth")
```

- Comparing characters is really comparing ASCII values of characters
- Comparing string objects is comparing the ASCII values of the characters in the strings. Comparison is character-by-character
- Cannot compare C-style strings with relational operators

4.11 The Conditional Operator

- Can use to create short **if/else** statements
- Format:

  expr ? expr : expr;

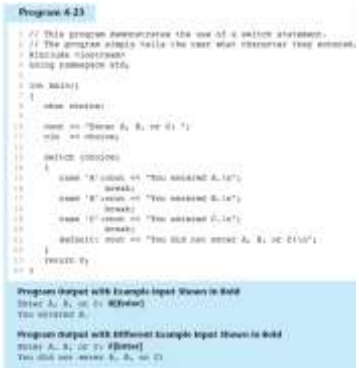


4.12 The **switch** Statement

- Used to select among statements from several alternatives
- May sometimes be used instead of **if/else if** statements

`switch` Statement Format

```
switch (IntExpression)
{
   case exp1: statement set 1;
   case exp2: statement set 2;
   ...
   case expn: statement set n;
   default:   statement set n+1;
}
```

`switch` Statement Requirements

1) ***IntExpression*** must be a **`char`** or an integer variable or an expression that evaluates to an integer value

2) ***exp1*** through ***expn*** must be constant integer type expressions and must be unique in the **`switch`** statement

3) **`default`** is optional but recommended

How the `switch` Statement Works

1) ***IntExpression*** is evaluated

2) The value of ***intExpression*** is compared against ***exp1*** through ***expn***.

3) If ***IntExpression*** matches value ***expi***, the program branches to the statement(s) following ***expi*** and continues to the end of the **`switch`**

4) If no matching value is found, the program branches to the statement after **`default:`**

The `break` Statement

- Used to stop execution in the current block
- Also used to exit a **`switch`** statement
- Useful to execute a single **`case`** statement without executing statements following it

Example `switch` Statement

```
switch (gender)
{
  case 'f': cout << "female";
            break;
  case 'm': cout << "male";
            break;
  default : cout << "invalid gender";
}
```


Using **switch** with a Menu

switch statement is a natural choice for menu-driven program

- display menu
- get user input
- use user input as **IntExpression** in **switch** statement
- use menu choices as **exp** to test against in the **case** statements

Program 4-27

```
// This menu-driven program uses a switch statement to carry out
// the appropriate set of actions based on the user's menu choice.
#include <iostream>
#include <iomanip>
using namespace std;

int main()
{
   // Constants for membership rates
   const double ADULT_RATE  = 40.0;
   const double CHILD_RATE  = 20.0;
   const double SENIOR_RATE = 30.0;

   int choice;          // Menu choice
   int months;          // Number of months
   double charges;      // Monthly charges

   // Display the menu and get the user's choice
   cout << " Health Club Membership Menu\n\n";
   cout << "1. Standard Adult Membership\n";
   cout << "2. Child Membership\n";
   cout << "3. Senior Citizen Membership\n";
   cout << "4. Quit the Program\n\n";
   cout << "Enter your choice: ";
   cin  >> choice;

   // Validate and process the menu choice
   if (choice < 1 || choice > 4)
   {  cout << "You must enter ";
      cin  >> months;
```



```
      // Set charges based on user input
      switch (choice)
      {
         case 1: charges = months * ADULT_RATE;
                 break;
         case 2: charges = months * CHILD_RATE;
                 break;
         case 3: charges = months * SENIOR_RATE;
      }
      // Display the monthly charges
      cout << fixed << showpoint << setprecision(2);
      cout << "The total charges are $" << charges << endl;
   }
   else if (choice != 4)
   {  cout << "The valid choices are 1 through 4.\n";
      cout << "Run the program again and select one of those.\n";
   }
   return 0;
}
```

```
 Health Club Membership Menu

1. Standard Adult Membership
2. Child Membership
3. Senior Citizen Membership
4. Quit the Program

Enter your choice: 2[Enter]
For how many months? 6[Enter]
The total charges are $120.00
```

Not for CSCI 2010

HW

Lab 4

1. Students should read the Pre-lab Reading Assignment before coming to lab.
2. Students should complete the Pre-lab Writing Assignment before coming to lab. (photocopy or copy/paste, answer then print and bring to class.)

For CSCI 2010

Prog2

Exercises Deitel Ch2 Intro to C++ Programming.pdf

Do the 5 programming exercises:

2.19 arithmetic.cpp

2.21 diamond.cpp

2.24 odd_even.cpp

2.25 multiples.cpp

2.28 digits.cpp (a bit challenging !)

4.13 Enumerated Data Types

- Data type created by programmer
- Contains a set of named constant integers
- Format:

```
enum name {val1, val2, … valn};
```

- Examples:

```
enum Fruit {apple, grape, orange};
enum Days {Mon, Tue, Wed, Thu, Fri, Sat, Sun};
```


Enumerated Data Type Variables

- To define variables, use the enumerated data type name

```
Fruit snack;
Days workDay, vacationDay;
```

- Variable may contain any valid value for the data type

```
snack = orange;      // no quotes
if (workDay == Wed) // none here
```


Enumerated Data Type Values

- Enumerated data type values are associated with integers, starting at 0

```
enum Fruit {apple, grape, orange};
```

- Can override default association

```
enum Fruit {apple = 2, grape = 4, orange = 5}
```

0 1 2

Avoid !

Enumerated Data Type Notes

- Enumerated data types improve the readability of a program
- Enumerated variables can not be used with input statements, such as **`cin`**
- Will not display the name associated with the value of an enumerated data type if used with **`cout`**