Tony Gaddis 5th Ed
Starting Out with C++

COMPUTER SCIENCE

CHAPTER 4
MAKING DECISIONS

Mohammad Al-Husseini

Relational Operators

- Thus far, your programs have asked for input, done some calculation, and displayed a result
- This is not an entirely useful program, its always going to perform the same operation
- We can compare pieces of data to perform more complex logic

Relational Operators

>	Greater Than
<	Less Than
>=	Greater Than or Equal To
<=	Less Than or Equal To
==	Equal To
!=	Not Equal To

x > y	Is x Greater Than y?
x < y	Is x Less Than y?
x >= y	Is x Greater Than or Equal To y?
x <= y	Is x Less Than or Equal To y?
x == y	Is x Equal To y?
x != y	Is x Not Equal To y?

Relational Operators

- The result of a relational expression is boolean (true or false)
- Recall that true and false are keywords
  - false represents the integer 0
  - true represents any non-zero integer, usually 1
    - They are technically constants and true = 1
    - But the logic of true can be any non-zero integer
- Warning: Don't confuse == and =
  - = is the assignment operator
  - == is the equality operator

Relational Operators

- Note: Relational operators have a higher precedence than the assignment operator
- Assume x = 10, y = 7, and z, a, and b are either ints or bools
  - z = x < y; z = 0 because x < y is false
  - a = x >= y; a = 1
  - b = y != x; b = 1
  - cout << (x > y); 1
  - cout << x <= y; 0

Checkpoint

1. Assume x = 5, y = 6, and z = 8. True or False
    1. x == 5
    2. 7 <= (x + 2)
    3. z < 4
    4. (2 + x) != y
    5. z != 4
    6. x >= 9
    7. x <= (y * 2)

Checkpoint

2. True or False
   1. x <= y is the same as y > x
   2. x != y is the same as y >= x
   3. x >= y is the same as y <= x
3. Yes or No
   1. If x > y and x < z, is y < z?
   2. If x >= y and z == x, is z == y?
   3. If x != y and x != z, is z != y?

Checkpoint

4. What will the following program display?

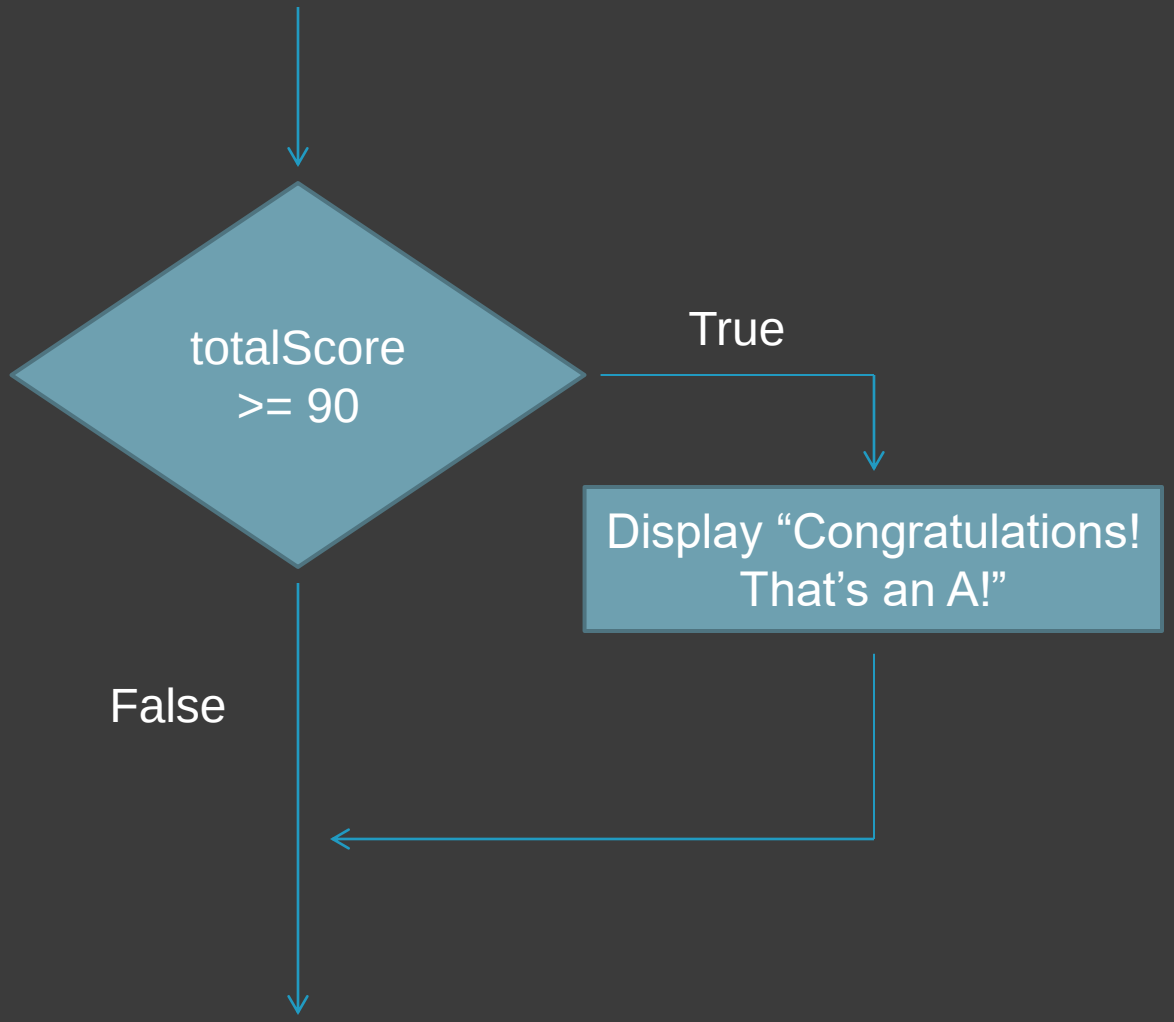
```
#include <iostream>
using namespace std;

int main()
{
    int a = 0, b = 2, x = 4, y = 0;

    cout << (a == b) << endl;
    cout << (a != y)  << endl;
    cout << (b <= x) << endl;
    cout << (y > a)   << endl;
    return 0;
}
```

The if Statement

- Previous programs have followed a sequence structure, taking one step after another
  - cout << “Enter a length: “;
    cin >> length;
    cout << “Enter a width: “;
    cin >> width;
    area = length * width;
    cout << “The area is: “ << area;
- But most programs perform more complex operations that require more than one sequential path
  - What if we wanted the area of other shapes?

The if Statement

- A simple example is a letter grade calculator



The if Statement

- if (expression)
  statement;

```
1  // This program determines a student's score, Only Main Body Shown
2
3     int score1, score2, score3;  // To hold three scores
4     double totalScore;           // To hold the total score
5
6     // Get the three scores.
7     cout << "Enter the 3 scores: ";
8     cin >> score1 >> score2 >> score3;
9
10    // Calculate and display the score.
11    totalScore = (score1 + score2 + score3) / 3.0;
12    cout << fixed << showpoint << setprecision(1);
13    cout << "Your score is " << totalScore << endl;
14
15    // If the score is greater than 89, congratulate the user.
16    if (totalScore >= 90)
17       cout << "Congratulations! That's an A!\n";
```

if Notes

- Do not put a semicolon after the if statement
  - bool b = false;
    if(b);
      cout << “b is true”;
  - This skips the if statement and the cout is always executed
- The statements in the if statement should be indented for readability
  - bool meeting = true;

```
if(meeting)                    if(meeting)
cout << “Hello”;                       cout << “Hello”
cout << “Goodbye.”             cout << “Goodbye.”
```

if Notes

- Do not mix up the assignment and equality operators
  - int a = 2, b = 3;
    if(a = b)
      cout << “A is the same as B”;
  - Remember that the assignment operator returns its value, so a will be set equal to b and its value returned
  - That value is non-zero so it will equate to true
  - And this will display that A is the same as B

IF NOTES

- Avoid the equality operator when comparing floating point numbers, use >= or <=

```
1  // This program demonstrates how floating-point
2  // round-off errors can make equality operations unreliable.
3
4     double a = 1.5;              // a is 1.5.
5     double b = 1.5;              // b is 1.5.
6
7     a += 0.0000000000000001;     // Add a little to a.
8     if (a == b)
9        cout << "Both a and b are the same.\n";
10    else
11       cout << "a and b are not the same.\n";
```

- Will display that they are the same
- This is behavior is caused by rounding errors

Expanding the if Statement

- You can have multiple statements inside the if
- To do so, you must use braces
- if(expression
  {
  statement;
  statement;
  …
  }
- You don’t need the braces with a single statement, but it is still good to include them for clarity

Expanding the if Statement

```
1  // This program averages 3 test scores.
2  // It demonstrates an if statement executing
3  // a block of statements.
4  #include <iostream>
5  #include <iomanip>
6  using namespace std;
7
8  int main()
9  {
10    int score1, score2, score3;  // To hold three test scores
11    double average;              // TO hold the average score
12
13    // Get the three test scores.
14    cout << "Enter 3 test scores and I will average them: ";
15    cin >> score1 >> score2 >> score3;
16
17    // Calculate and display the average score.
18    average = (score1 + score2 + score3) / 3.0;
19    cout << fixed << showpoint << setprecision(1);
20    cout << "Your average is " << average << endl;
21
22    // If the average is greater than 95, congratulate the user.
23    if (average > 95)
24    {
25       cout << "Congratulations!\n";
26       cout << "That's a high score.\n";
27       cout << "You deserve a pat on the back!\n";
28    }
29    return 0;
30 }
```

Don't Forget the Braces

```
1  // This program averages 3 test scores. The braces
2  // were inadvertently left out of the if statement.
3  #include <iostream>
4  #include <iomanip>
5  using namespace std;
6
7  int main()
8  {
9     int score1, score2, score3;  // To hold three test scores
10    double average;              // TO hold the average score
11
12    // Get the three test scores.
13    cout << "Enter 3 test scores and I will average them: ";
14    cin >> score1 >> score2 >> score3;
15
16    // Calculate and display the average score.
17    average = (score1 + score2 + score3) / 3.0;
18    cout << fixed << showpoint << setprecision(1);
19    cout << "Your average is " << average << endl;
20
21    // ERROR! This if statement is missing its braces!
22    if (average > 95)
23       cout << "Congratulations!\n";
24       cout << "That's a high score.\n";
25       cout << "You deserve a pat on the back!\n";
26    return 0;
27 }
```

Flag Variables

- Typically a bool variable that signals when a condition is met
  - bool highScore = false;
    if (average > 95)
      highScore = true;
    if(highscore)
      cout << “Congratulations! That’s a high score!”;
- Can also be integer
  - bool highScore = 0; // 0 representing false
  - You must initialize these variables

Checkpoint

5. True or False, These are equivalent

```
if(sales > 10000)
    commissionRate = 0.15;

if(sales > 10000) commissionRate = 0.15;
```

6. True or False, These are equivalent

```
if(calls == 20)
    rate *= 0.5;

if(calls = 20)
    rate *= 0.5;
```

Checkpoint

7. Each contains a logic error
   Assume the variables exist
   1. if(hours > 40);
      cout << hours << " hours qualifies for over-time.";

   2. balance = 1000;
      if(interestRate = .07)
      cout << "This account is earning the maximum rate.";

   3. if(interestRate >.07)
      cout << "This account earns a $10 bonus.";
      balance += 10.0;

Checkpoint

8. Write an if statement that assigns 0 to x if y is equal to 20
9. Write an if statement that multiples payRate by 1.5 if hours is greater than 40
10. Write an if statement that assigns .20 to comission if sales is greater than or equal to 10000.00
11. Write an if statement that sets the variable fees to 50 if the variable max is set to true

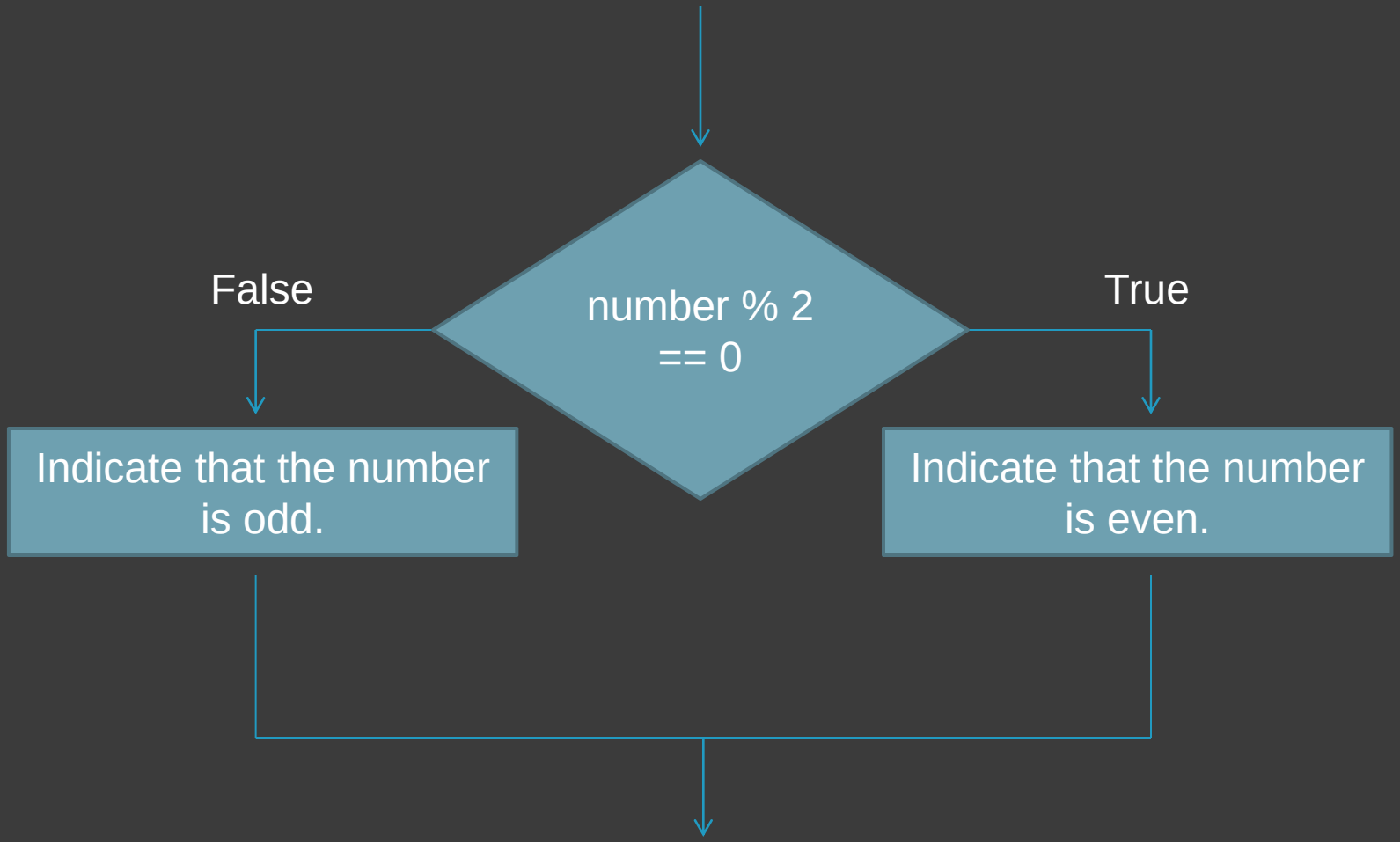
The if/else Statement

- Executes the first block if true, else it executes the second block
- if (expression)
  statement or block
  else
  statement or block
- Useful for many things, especially avoiding division by zero

```
cout << “Enter two numbers: “
cin >> a >> b;
if(b != 0)
    cout << a / b;
```

The if/else Statement

- Determine if a number is even or odd



The if/else Statement

```
1  // This program uses the modulus operator to determine
2  // if a number is odd or even. If the number is evenly divisible
3  // by 2, it is an even number. A remainder indicates it is odd.
4  #include <iostream>
5  using namespace std;
6
7  int main()
8  {
9     int number;
10
11    cout << "Enter an integer and I will tell you if it\n";
12    cout << "is odd or even. ";
13    cin >> number;
14    if (number % 2 == 0)
15       cout << number << " is even.\n";
16    else
17       cout << number << " is odd.\n";
18    return 0;
19 }
```

Checkpoint

12. True of False, These are equivalent

1.
```
if(x > y)
   cout << “x is the greater”;
else
   cout << “x is not the greater.”;
```

2.
```
if(y <= x)
   cout << “x is not the greater.”;
else
   cout << “x is the greater.”;
```

Checkpoint

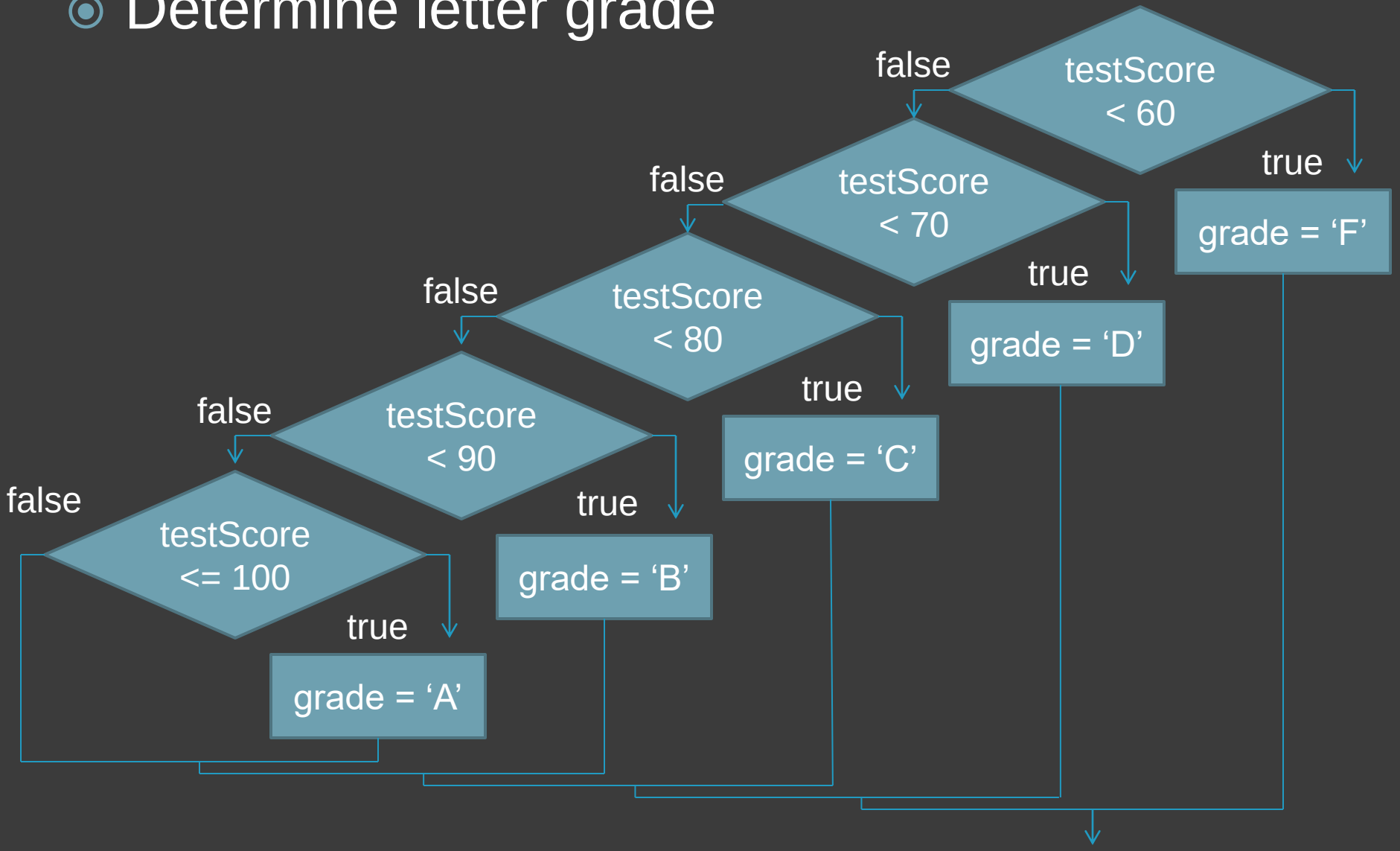
13. Write an if/else statement that assigns 1 to x if y is equal to 100. Otherwise it should assign 0 to x
14. Write an if/else statement that assigns 0.10 to commission unless sales is greater than or equal to 50000.00, in which case it assigns 0.20 to commission
15. Write an if/else statement that assigns true to the variable even if n % 2 is true. Otherwise it should assign false to even.

The if/else if Statement

- Starts at the first block, executing if true
  Otherwise continues down each block until one is true (can be multiple else if blocks)
  Otherwise executes the else block
  Only executes a single block (the first true)
- if (expression)
  statement or block
  else if (expression)
  statement or block
  else
  statement or block

The if/ese if Statement

- Determine letter grade



The if/else if Statement

```
1  // This program uses an if/else if statement to assign a
2  // letter grade (A, B, C, D, or F) to a numeric test score.
3     int testScore;  // To hold a numeric test score
4     char grade;     // To hold a letter grade
5
6     // Get the numeric test score.
7     cout << "Enter your numeric test score and I will\n";
8     cout << "tell you the letter grade you earned: ";
9     cin >> testScore;
10
11    // Determine the letter grade.
12    if (testScore < 60)
13       grade = 'F';
14    else if (testScore < 70)
15       grade = 'D';
16    else if (testScore < 80)
17       grade = 'C';
18    else if (testScore < 90)
19       grade = 'B';
20    else if (testScore <= 100)
21       grade = 'A';
22
23    // Display the letter grade.
24    cout << "Your grade is " << grade << ".\n";
```

Using a Trailing else

- But what happens when the user enters a value greater than 100?
- There is nothing to catch that event, so we've created a crack through which the program can fall
  - grade will never been initialized and so the code will throw an error when it attempts to access that value
- Include an else at the end to catch all other possible values and prevent such things

Using a Trailing else

```
1  // This program uses an if/else if statement to assign a
2  // letter grade (A, B, C, D, or F) to a numeric test score.
3     int testScore;  // To hold a numeric test score
4     char grade;     // To hold a letter grade
5
6     // Get the numeric test score.
7     cout << "Enter your numeric test score and I will\n";
8     cout << "tell you the letter grade you earned: ";
9     cin >> testScore;
10
11    // Display the letter grade.
12    if (testScore < 60)
13       cout << "Your grade is F.\n";
14    else if (testScore < 70)
15       cout << "Your grade is D.\n";
16    else if (testScore < 80)
17       cout << "Your grade is C.\n";
18    else if (testScore < 90)
19       cout << "Your grade is B.\n";
20    else if (testScore <= 100)
21       cout << "Your grade is A.\n";
22    else
23       cout << "We do not give scores higher than 100.\n";
```

Multiple if vs if/else if

- You can use multiple if statements in conjunction to achieve a similar effect
- But keep in mind the advantage of if/else if is that it stops executing after the first true
- Multiple if statements could execute multiple times

Multiple if vs if/else if

```
1  // This program uses independent if/else statements to assign a
2  // letter grade (A, B, C, D, or F) to a numeric test score.
3  // Do you think it will work?
4     int testScore;  // To hold a numeric test score
5     char grade;     // To hold a letter grade
6
7     // Get the numeric test score.
8     cout << "Enter your numeric test score and I will\n";
9     cout << "tell you the letter grade you earned: ";
10    cin >> testScore;
11
12    // What letter grade will be assigned?
13    if (testScore < 60)
14       grade = 'F';
15    if (testScore < 70)
16       grade = 'D';
17    if (testScore < 80)
18       grade = 'C';
19    if (testScore < 90)
20       grade = 'B';
21    if (testScore <= 100)
22       grade = 'A';
23
24    // Display the letter grade.
25    cout << "Your grade is " << grade << ".\n";
```

Menus

```
1  // This program displays a menu and asks the user to make a
2  // selection. An if/else if statement determines which item
3  // the user has chosen.
4  #include <iostream>
5  #include <iomanip>
6  using namespace std;
7
8  int main()
9  {
10    int choice;         // Menu choice
11    int months;         // Number of months
12    double charges;     // Monthly charges
13
14    // Constants for membership rates
15    const double ADULT = 40.0;
16    const double SENIOR = 30.0;
17    const double CHILD = 20.0;
18
19    // Display the menu and get a choice.
20    cout << "\t\tHealth Club Membership Menu\n\n";
21    cout << "1. Standard Adult Membership\n";
22    cout << "2. Child Membership\n";
23    cout << "3. Senior Citizen Membership\n";
24    cout << "4. Quit the Program\n\n";
25    cout << "Enter your choice: ";
26    cin >> choice;
```

Menus

```
28      // Set the numeric ouput formatting.
29      cout << fixed << showpoint << setprecision(2);
30
31      // Respond to the user's menu selection.
32      if (choice == 1)
33      {
34          cout << "For how many months? ";
35          cin >> months;
36          charges = months * ADULT;
37          cout << "The total charges are $" << charges << endl;
38      }
39      else if (choice == 2)
40      {
41          cout << "For how many months? ";
42          cin >> months;
43          charges = months * CHILD;
44          cout << "The total charges are $" << charges << endl;
45      }
```

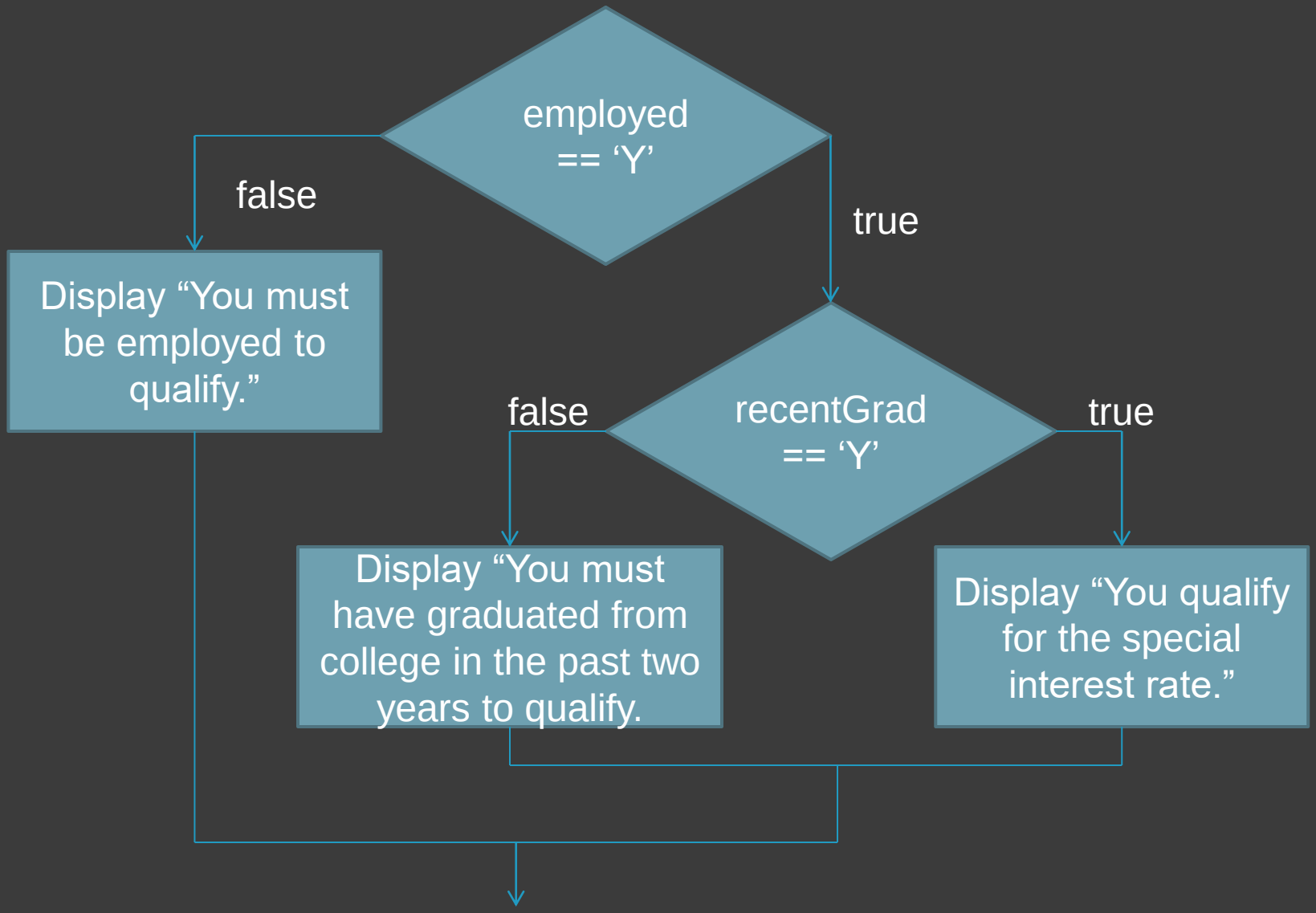
Menus

```
46      else if (choice == 3)
47      {
48          cout << "For how many months? ";
49          cin >> months;
50          charges = months * SENIOR;
51          cout << "The total charges are $" << charges << endl;
52      }
53      else if (choice != 4)
54      {
55          cout << "The valid choices are 1 through 4. Run the\n";
56          cout << "program again and select one of those.\n";
57      }
58      return 0;
59  }
```

Nested if Statements

- You can put if statements in the block of other if statements
- This allows for very meaningful and deep logic
  - If it is raining outside and it is windy take a jacket
  - If it is raining outside and it is calm take a umbrella

Nested if Statements



Nested if Statements

```
1  // This program demonstrates the nested if statement.
2  #include <iostream>
3  using namespace std;
4
5  int main()
6  {
7     char employed,     // Currently employed, Y or N
8          recentGrad;   // Recent graduate, Y or N
9
10    // Is the user employed and a recent graduate?
11    cout << "Answer the following questions\n";
12    cout << "with either Y for Yes or ";
13    cout << "N for No.\n";
14    cout << "Are you employed? ";
15    cin >> employed;
16    cout << "Have you graduated from college ";
17    cout << "in the past two years? ";
18    cin >> recentGrad;
19
```

Nested if Statements

```
20      // Determine the user's loan qualifications.
21      if (employed == 'Y')
22      {
23          if (recentGrad == 'Y') // Nested if
24          {
25              cout << "You qualify for the special ";
26              cout << "interest rate.\n";
27          }
28          else // Not a recent grad, but employed
29          {
30              cout << "You must have graduated from ";
31              cout << "college in the past two\n";
32              cout << "years to qualify.\n";
33          }
34      }
35      else  // Not employed
36      {
37          cout << "You must be employed to qualify.\n";
38      }
39      return 0;
40  }
```

Checkpoint

16.
```
int numBooks, numCoupons;
cout << “How many books are being purchased? “;
cin >> numBooks;
if (numBooks < 1)
    numCoupons = 0;
else if (numBooks < 3)
    numCoupons = 1;
else if (numBooks <5)
    numCoupons = 2;
else
    numCoupons = 3;
cout << “The number of coupons is “ numCoupons;
```

User Enters	Coupons
1	?
2	
3	
4	
5	
10	

Checkpoint

17. Write nested if statements that perform the following test:

    If amount1 is greater than 10 and amount2 is less than 100, display the greater of the two.

18. What is the difference between multiple if statements, if/else if statements, and nested if statements?

Logical Operators

- Generally connect two or more relational expressions to further enhance the logic
- AND
- OR
- NOT

Logical Operators

- AND
  - &&
  - Both expressions must be true for the result to be true

Left	Right	Result
0	0	0
0	1	0
1	0	0
1	1	1

  - Short circuit evaluation – If the left expression is false, then it doesn't waste CPU time checking the right expression (result is false)

Logical Operators

- OR
  - ||
  - Either expression (or both) must be true for the result to be true

Left	Right	Result
0	0	0
0	1	1
1	0	1
1	1	1

  - Short circuit evaluation – If the left expression is true, then it doesn't waste CPU time checking the right expression (result is true)

Logical Operators

- NOT
  - !
  - Reverses the value of the expression (makes false true or true false)
  - Unary

Expression	Result
0	1
1	0

Logical Operators

- Precedence
  - !
  - &&
  - ||
  - Note: AND and OR have lower precedence than the relational operators
  - Note: But NOT has a higher precedence
    - !(x > 2) // if x is greater than 2 return false
    - !x > 2 // is the logical negation of x greater than 2?
      - Suppose x = 5. x is non-zero, so negating it makes it zero zero is not greater than 2, so the result is false

Logical Operators

- Examples:
  - if (employed == 'Y' && recentGrad == 'Y')
    cout << "You qualify for the special interest rate";
  - if (income >= 35000 || years > 5)
    cout << "You qualify for the loan";
  - if (!(income >= 35000 || years > 5))
    cout << "You must earn at least $35,000 or ";
    cout << "have been employed at least 5 years.";

Checking Numeric Ranges

- Logical operators allow us to verify is a number is inside or outside of a range
  - Use AND to check if it is inside a range
    - if (x >= 20 && x <= 40)
      cout << “x is 20 – 40)”;
  - Use OR to check if it is outside a range
    - if (x < 20 || x > 40)
      cout << “x is not 20 – 40”;
  - Beware
    - if (x < 20 && x > 40)
    - if (x < 20 | x > 40)
    - if (x < 20 || > 40)

if (x >=20 & x <= 40)

if (20 < x < 40)

Checkpoint

19. What is the disadvantage of using the && logical operator instead of the nested if?

20.

Expression	Result
true && false	?
true && true	
false && true	
false && false	
true || false	
true || true	
false || true	
false || false	
!true	
!false	

Checkpoint

21. Assume a = 2, b = 4, and c = 6. True or False
    1. a == 4 || b > 2
    2. 6 <= c && a > 3
    3. 1 != b && c != 3
    4. a >= -1 || a <= b
    5. !(a > 2)

Checkpoint

22. Write an if statement that prints the message “The number is valid” if the variable speed is within the range 0 through 200.
23. Write an if statement that prints the message “The number is not valid” if the variable speed is outside the range 0 through 200.

Validating User Input

- Never assume that the user enters the correct information, C++ does
- If you fail to do so, it could cause your program to crash and potentially lose data (if you are writing files)
- Use if statements in conjunction with relational and logical expressions

Validating User Input

```
11      // Get the numeric test score.
12      cout << "Enter your numeric test score and I will\n";
13      cout << "tell you the letter grade you earned: ";
14      cin >> testScore;
15
16      if (testScore < 0 || testScore > 100) //Input validation
17      {
18          // An invalid score was entered.
19          cout << testScore << " is an invalid score.\n";
20          cout << "Run the program again and enter a value\n";
21          cout << "in the range of 0 to 100.\n";
22      }
23      else
24      {
25          // Determine the letter grade.
26          if (testScore < 60)
27              grade = 'F';
28          else if (testScore < 70)
29              grade = 'D';
30          else if (testScore < 80)
31              grade = 'C';
32          else if (testScore < 90)
33              grade = 'B';
34          else if (testScore <= 100)
35              grade = 'A';
36
37          // Display the letter grade.
38          cout << "Your grade is " << grade << endl;
39      }
40      return 0;
```

Variable Scope

- A variable's scope is the block it is defined in, between a set of { }
  - This includes if statements
- It is generally best to define variables at the start of the block, but is sometimes better to define them (before) where they are relevant
- As long as two variables are in different scopes, they can have the same name (though you should generally avoid this)
  - Statements will use the variable in the same block as them

Variable Scope

```
1  // This program uses two variables with the name number.
2  #include <iostream>
3  using namespace std;
4
5  int main()
6  {
7     // Define a variable named number.
8     int number;
9
10    cout << "Enter a number greater than 0: ";
11    cin >> number;
12    if (number > 0)
13    {
14       int number;  // Another variable named number.
15       cout << "Now enter another number: ";
16       cin >> number;
17       cout << "The second number you entered was ";
18       cout << number << endl;
19    }
20    cout << "Your first number was " << number << endl;
21    return 0;
22 }
```

Checkpoint

24. What is wrong with this code:

```
int first, second, result;

cout << “Enter a negative integer: “;
cin >> first;
cout << “Enter a positive integer: “;
cin >> second;

if(first >= 0 || second < 0)
    cout << “Good job!”;
```

Checkpoint

25. What will the following display:

```
int test1 = 40, test2 = 30;
int sum = test1 + test2;

if(sum > 50)
{
    test1 += 10;
    test2 += 10;
    int sum = test1 + test2;
}

cout << test1 << “, “ << test2 << “, “ << sum;
```

Comparing Strings

- You cannot use relational operators to compare c-strings
  - if(firstString == secondString)
  - This compares the memory addresses, which are (almost) always different
- You must use a function
  - strcmp(string1, string)
  - Requires #include <cstring>
  - Returns 0 if the strings are the same
  - Returns <0 if string1 < string2
  - Returns >0 if string1 > string2

Comparing Strings

```
1  // This program uses strcmp to compare the string entered
2  // by the user with the valid stereo part numbers.
3  #include <iostream>
4  #include <iomanip>
5  #include <cstring>
6  using namespace std;
7
8  int main()
9  {
10    const double APRICE = 249.0, // Price A
11                 BPRICE = 299.0; // Price B
12    const int SIZE = 9;          // Array size
13    char partNum[SIZE];          // To hold the part number
14
15    // Get a part number from the user.
16    cout << "The stereo part numbers are:\n";
17    cout << "\tBoom Box, part number S147-29A\n";
18    cout << "\tShelf Model, part number S147-29B\n";
19    cout << "Enter the part number of the stereo you\n";
20    cout << "wish to purchase: ";
21    cin.width(SIZE);  // Restrict input for code safety.
22    cin >> partNum;
```

Comparing Strings

```
24      // Set the numeric output formatting.
25      cout << fixed << showpoint << setprecision(2);
26
27      // Determine and display the correct price.
28      if (strcmp(partNum, "S147-29A") == 0)
29          cout << "The price is $" << APRICE << endl;
30      else if (strcmp(partNum, "S147-29B") == 0)
31          cout << "The price is $" << BPRICE << endl;
32      else
33          cout << partNum << " is not a valid part number.\n";
34      return 0;
35  }
```

- Think of strcmp as reverse logic
  - if (!strcmp(partNum, “S147-29A”))

    else if (!strcmp(partNum, “S147-29B”))

Sorting Strings

- Utilize the negative or positive return from strcmp to determine the order
- If string1 > string2, the result is positive
  - This means string1's ASCII representation is a larger number, which means it appears later in the alphabet
  - Therefore, this is comes after string2 alphabetically
  - Remember the order
    - Numbers
    - Uppercase
    - Lowercase

Sorting Strings

```
1  // This program uses the return value of strcmp to alphabetically
2  // sort two strings entered by the user.
3  #include <iostream>
4  #include <cstring>
5  using namespace std;
6
7  int main()
8  {
9     const int SIZE = 30;
10    char name1[SIZE], name2[SIZE];
11
12    // Get the first name.
13    cout << "Enter a name (last name first): ";
14    cin.getline(name1, SIZE);
15
16    // Get the second name.
17    cout << "Enter another name: ";
18    cin.getline(name2, SIZE);
19
20    // Display them sorted in alphabetical order.
21    cout << "Here are the names sorted alphabetically:\n";
22    if (strcmp(name1, name2) < 0)
23       cout << name1 << endl << name2 << endl;
24    else if (strcmp(name1, name2) > 0)
25       cout << name2 << endl << name1 << endl;
26    else
27       cout << "You entered the same name twice!\n";
28    return 0;
29 }
```

Checkpoint

26. Indicate 0, positive, or negative
    1. strcmp(“ABC”, “abc”);
    2. strcmp(“Jill”, “Jim”);
    3. strcmp(“123”, “ABC”);
    4. strcmp(“Sammy”, “Sally”);

27. Would you use strcmp to compare characters?

The Conditional Operator

- The only ternary operator, which is a shorthand for the if/else statement
- expression ? expression : expression;
- test ? true : false;
- x < 0 ? y = 10 : z = 20;
  - If x < 0, then y = 10, else z = 20
- It returns its value
  - a = x > 100 ? 0 : 1;
  - If x > 100, then a = 0, else a = 1

```
if (x > 100)
    a = 0;
else
    a = 1;
```

The Conditional Operator

```
1  // This program calculates a consultant's charges at $50
2  // per hour, for a minimum of 5 hours. The ?: operator
3  // adjusts hours to 5 if less than 5 hours were worked.
4  #include <iostream>
5  #include <iomanip>
6  using namespace std;
7
8  int main()
9  {
10    const double PAY_RATE = 50.0;
11    double hours, charges;
12
13    cout << "How many hours were worked? ";
14    cin >> hours;
15    hours = hours < 5 ? 5 : hours; //conditional operator
16    charges = PAY_RATE * hours;
17    cout << fixed << showpoint << setprecision(2);
18    cout << "The charges are $" << charges << endl;
19    return 0;
20 }
```

The Conditional Operator

- Be warned, it had a lower precedence than the stream operators
  - cout << “Your grade is: “
    << (score < 60 ? “Fail.” : “Pass”);
  - Hence it requires the parenthesis

Checkpoint

28. Rewrite the if/else statements as a conditional expressions

    1.
```
if (hours > 40)
   wages *= 1.5;
else
   wages *= 1;
```

    2.
```
if (result >= 0)
   cout << “The result is positive\n”;
else
   cout << “The result is negative.\n”;
```

Checkpoint

29. Rewrite the conditional expresses as if/else statements
    1. total += count == 1 ? sales : count * sales;

    2. cout << (((num % 2) == 0) ? “Even” : “Odd”);

The switch Statement

- Like an if / else if / else statement, but you are comparing a single integer/character variable with multiple values
- Excellent for menus
- switch (IntegerExpression)

```
{
   case ConstantExpression:
      // One or more statements
   …
   default:
      // One or more statements
}
```

The switch Statement

- Your switch must be an integer or character variable (generally)
- Your cases must be a constant and cannot be an expression or variable
- You may have many cases but they must all be unique
- The default statement is optional
- Use break; between your cases to prevent fall-through
- break is optional for the default statement

The switch Statement

```
1  // The switch statement in this program tells the user something
2  // he or she already knows: what they just entered!
3  #include <iostream>
4  using namespace std;
5
6  int main()
7  {
8     char choice;
9
10    cout << "Enter A, B, or C: ";
11    cin >> choice;
12    switch (choice)
13    {
14       case 'A': cout << "You entered A.\n";
15                 break;
16       case 'B': cout << "You entered B.\n";
17                 break;
18       case 'C': cout << "You entered C.\n";
19                 break;
20       default:  cout << "You did not enter A, B, or C!\n";
21    }
22    return 0;
23 }
```

The switch Statement

- If you do not use break; then it will continue executing lines
- This is a confusing error
- But sometimes, this is also a behavior you want

Accidental Fallthrough

```
1  // The switch statement in this program tells the user something
2  // he or she already knows: what they just entered!
3  #include <iostream>
4  using namespace std;
5
6  int main()
7  {
8     char choice;
9
10    cout << "Enter A, B, or C: ";
11    cin >> choice;
12    // The following switch is
13    // missing its break statements!
14    switch (choice)
15    {
16       case 'A': cout << "You entered A.\n";
17       case 'B': cout << "You entered B.\n";
18       case 'C': cout << "You entered C.\n";
19       default:  cout << "You did not enter A, B, or C!\n";
20    }
21    return 0;
22 }
```

Intentional Fallthrough

```
1  // This program is carefully constructed to use the "fallthrough"
2  // feature of the switch statement.
3  #include <iostream>
4  using namespace std;
5
6  int main()
7  {
8     int modelNum;  // Model number
9
10    // Get a model number from the user.
11    cout << "Our TVs come in three models:\n";
12    cout << "The 100, 200, and 300. Which do you want? ";
13    cin >> modelNum;
14
15    // Display the model's features.
16    cout << "That model has the following features:\n";
17    switch (modelNum)
18    {
19       case 300: cout << "\tPicture-in-a-picture.\n";
20       case 200: cout << "\tStereo sound.\n";
21       case 100: cout << "\tRemote control.\n";
22                 break;
23       default:  cout << "You can only choose the 100,";
24                 cout << "200, or 300.\n";
25    }
26    return 0;
27 }
```

Intentional Fallthrough

```
1  // The switch statement in this program uses the "fall through"
2  // feature to catch both uppercase and lowercase letters entered
3  // by the user.
4  #include <iostream>
5  using namespace std;
6
7  int main()
8  {
9     char feedGrade;
10
11    // Get the desired grade of feed.
12    cout << "Our dog food is available in three grades:\n";
13    cout << "A, B, and C. Which do you want pricing for? ";
14    cin >> feedGrade;
15
16    // Display the price.
17    switch(feedGrade)
18    {
19       case 'a':
20       case 'A': cout << "30 cents per pound.\n";
21                 break;
22       case 'b':
23       case 'B': cout << "20 cents per pound.\n";
24                 break;
25       case 'c':
26       case 'C': cout << "15 cents per pound.\n";
27                 break;
28       default:  cout << "That is an invalid choice.\n";
29    }
30    return 0;
31 }
```

Menu switch

```
1  // This program displays a menu and asks the user to make a
2  // selection. A switch statement determines which item
3  // the user has chosen.
4  #include <iostream>
5  #include <iomanip>
6  using namespace std;
7
8  int main()
9  {
10    int choice;        // Menu choice
11    int months;        // Number of months
12    double charges;    // Monthly charges
13
14    // Constants for membership rates
15    const double ADULT = 40.0;
16    const double SENIOR = 30.0;
17    const double CHILD = 20.0;
18
19    // Display the menu and get a choice.
20    cout << "\t\tHealth Club Membership Menu\n\n";
21    cout << "1. Standard Adult Membership\n";
22    cout << "2. Child Membership\n";
23    cout << "3. Senior Citizen Membership\n";
24    cout << "4. Quit the Program\n\n";
25    cout << "Enter your choice: ";
26    cin >> choice;
```

Menu switch

```
28      // Validate and process the menu choice.
29      if (choice >= 1 && choice <= 3)
30      {
31          // Get the number of months.
32          cout << "For how many months? ";
33          cin >> months;
34
35          // Set the numeric output formatting.
36          cout << fixed << showpoint << setprecision(2);
37
38          // Respond to the user's menu selection.
39          switch (choice)
40          {
41              case 1:  charges = months * ADULT;
42                       break;
43              case 2:  charges = months * CHILD;
44                       break;
45              case 3:  charges = months * SENIOR;
46          }
47
48          // Display the monthly charges.
49          cout << "The total charges are $";
50          cout << charges << endl;
51      }
52      else if (choice != 4)
53      {
54          cout << "The valid choices are 1 through 4. Run the\n";
55          cout << "program again and select one of those.\n";
56      }
57      return 0;
58  }
```

Checkpoint

30. Why can't you convert this isn't a switch?

```
if (temp == 100)
   x = 0;
else if (population > 1000)
   x = 1;
else if (rate < .1)
   x = -1;
```

Checkpoint

31. What is wrong with this switch statement?

```
switch (temp)
{
   case temp < 0   : cout << “Temp is negative.”;
                     break;
   case temp == 0 : cout << “Temp is zero.”;
                     break;
   case temp > 0   : cout << “Temp is positive.”;
                     break;
}
```

Testing for File Open Errors

- Recall that input files are not created when opened
  - This will cause an error if they do not exist
  - ifstream inputFile;
    inputFile.open(“info.txt”);
- Now that you know if statements, we can avoid this
- When the file is opened, its variable is set to either true (if it was opened successfully) or false (if it was not)

Testing for File Open Errors

- Use this true/false attribute to test the opening
  - ifstream inputFile;
    inputFile.open("info.txt");
    if(!inputFile)
    {
       cout << "Error opening file.";
    }
- You can also use a member function
  - ifstream inputFile;
    inputFile.open("info.txt");
    if(inputFile.fail())
       cout << "Error opening file.";

Testing for File Open Errors

- You can also check that output files opened
- Even though they are created when opened, there are still circumstances that could prevent them from being opened
  - ofstream outputFile;
    outputFile.open("customer.txt");
    if(outputFile.fail())
        cout << "The customer.txt file could not be opened. Perhaps the disk is full or you do not have sufficent privileges. Contact your system manager for assistance."