

# If/else & switch

## Unit 3

Sections 4.1-6, 4.8-12, 4.14-15

CS 1428  
Fall 2019

Jill Seaman

1

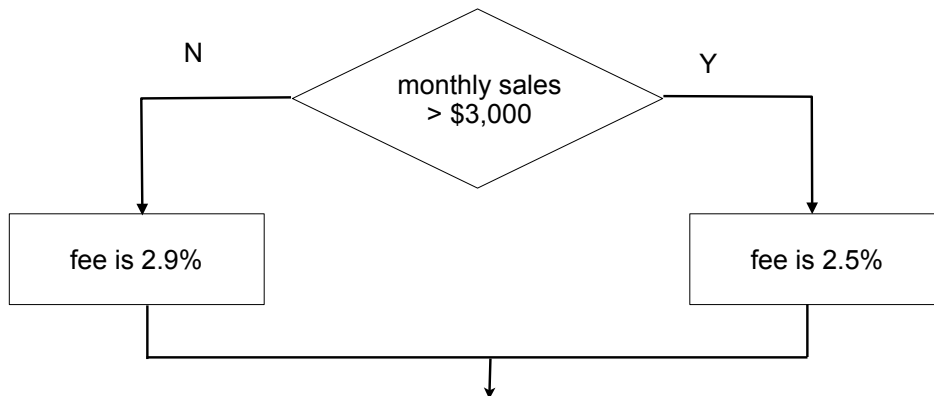
# Straight-line code (or IPO: Input-Process-Output)

- So far all of our programs have followed this basic format:
  - ▶ Input some values
  - ▶ Do some computations
  - ▶ Output the results
- The statements are executed in a sequence, first to last.

2

# Decisions

- Sometimes we want to be able to decide which of two statements to execute:



3

# Relational Expressions

- Making decisions require being able to ask “Yes” or “No” questions.
- Relational expressions allow us to do this.
- Relational expressions evaluate to **true** or **false**.
- Also called:
  - ▶ logical expressions
  - ▶ conditional expressions
  - ▶ boolean expressions

4

## Relational Expressions

- Boolean literals:

true

false

true evaluates to true

false evaluates to false

- Boolean variables

```
bool isPositive = true;
bool found = false;
```

isPositive evaluates to true  
found evaluates to false

5

## 4.1 Relational Operators

- Binary operators used to compare expressions:

< Less than

<= Less than or equal to

> Greater than

>= Greater than or equal to

== Equals (note: do not use =) !!

!= Not Equals

6

## Relational Expressions

- Examples:

```
int x=6;
int y=10;
```

a. `x == 5` evaluates to \_\_false\_\_  
b. `7 <= x + 2` evaluates to \_\_\_\_\_  
c. `y - 3 > x` evaluates to \_\_\_\_\_  
d. `x != y` evaluates to \_\_\_\_\_  
e. `true` evaluates to \_\_true\_\_

- Can assign relational expressions to variables:

```
bool isPositive;
int x;
cin >> x;
isPositive = x > 0;
```

if the user types: 25  
isPositive stores the value \_\_\_\_\_

7

## Relational Operator Precedence

- Relational operators are LOWER than arithmetic operators:

```
int x, y;

... x < y - 10 ... // minus happens first
... x * 5 >= y + 10 ... // mult, then plus, then >=
```

- Relational operators are HIGHER than assignment:

```
int x, y;
...
bool t1 = x > 7; // > then =
bool t2 = x * 5 >= y + 10; // *, +, >=, =
```

8

## 4.2 The if statement

- The if statement can be used to execute a statement only under certain conditions:

```
if (expression)
    statement
```

- **expression is evaluated**
  - ▶ If it is true, then `statement` is executed.
  - ▶ If it is false, then `statement` is skipped

9

## if statement example

- Example: An employee gets a \$100 bonus if their hours are over 40.

```
double rate = 14.50;
double hours, pay;

cout << "Enter the hours you worked: ";
cin >> hours;

pay = hours * rate;
if (hours > 40)
    pay = pay + 100;

cout << "Your pay is: $" << pay << endl;
```

10

## 4.3 The block statement

- a block (or a compound statement) is a set of statements inside braces:

```
{
    int x;
    cout << "Enter a value for x: " << endl;
    cin >> x;
    cout << "Thank you." << endl;
}
```

- This groups several statements into a single statement.
- This allows us to use multiple statements when by rule only one is allowed.

11

## if with a block

- We can use a block to conditionally execute more than just one statement:

```
double rate = 14.50;
double hours, pay;

cout << "Enter the hours you worked: ";
cin >> hours;

pay = hours * rate;
if (hours > 40) {
    pay = pay + 100;
    cout << "Your pay includes a bonus." << endl;
}

cout << "Your pay is: $" << pay << endl;
```

12

## 4.4 The if/else statement

- if/else statement is used to decide which of two statements to execute:

```
if (expression)
    statement1 (or block)
else
    statement2 (or block)
```

statement1 and statement2  
are called branches

- expression is evaluated
  - If it is true, then statement1 is executed.  
(statement2 is skipped).
  - If it is false, then statement2 is executed  
(statement1 is skipped).

13

## if/else example

```
double monthlySales;
double price;
double rate;

cout << "Enter monthly sales last month: " ;
cin >> monthlySales;
cout << "Enter selling price of item: " ;
cin >> price;

if (monthlySales > 3000)
    rate = .025;
else
    rate = .029;

double commission = price * rate;
cout << "Commission: $" << commission << endl;
```

```
Enter monthly sales last month: 3025
Enter selling price of item: 100
Commission: $2.50
```

14

## if/else structure

Notice:

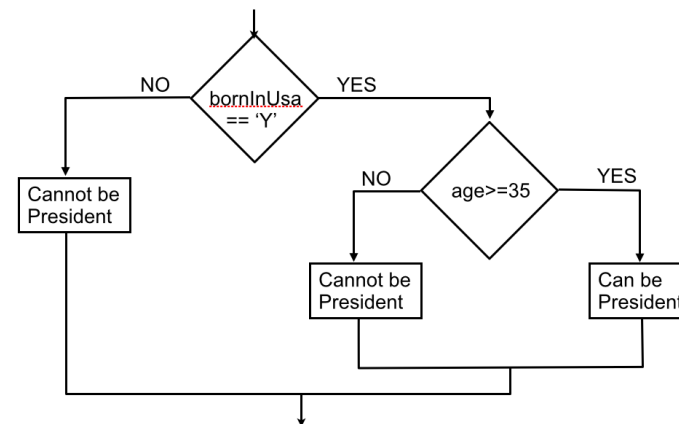
```
if (monthlySales > 3000)
    rate = .025;
else
    rate = .029;
```

- relational expression is in parentheses
- NO semi-colon after expression, nor the else
- Good style: indent the statements in each branch!!

15

## 4.5 Nested if statements

- if-else is a statement. It can occur as a branch of another if-else statement.



16

## Nested if statements

- if-else is a statement. It can occur as a branch of another if-else statement.

```
char bornInUSA;
int age;
cout << "Were you born in the USA (Y/N)?: " ;
cin >> bornInUSA;
cout << "Please enter your age: ";
cin >> age;

if (bornInUSA == 'Y')
    if (age >= 35)
        cout << "You qualify to run for President\n";
    else
        cout << "You are too young to run for President\n";
else
    cout << "You must have been born in the US in order "
        << "to run for President" << endl;
```

17

## Nested if statements

- if-else is a statement. It can occur as a branch of another if-else statement.

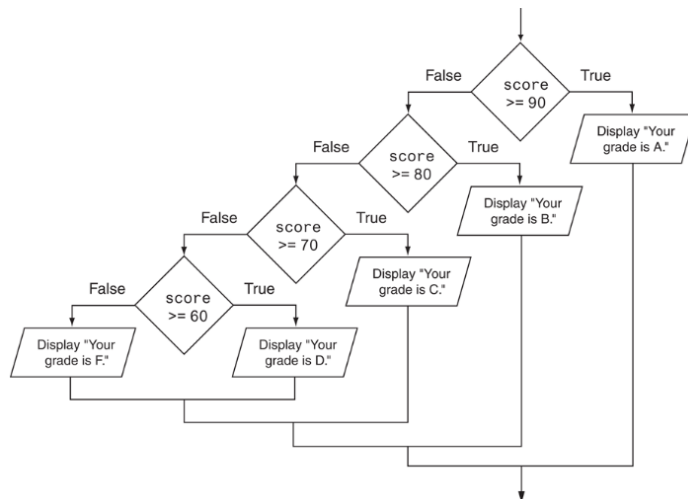
```
char bornInUSA;
int age;
cout << "Were you born in the USA (Y/N)?: " ;
cin >> bornInUSA;
cout << "Please enter your age: ";
cin >> age;

if (bornInUSA == 'Y')
    if (age >= 35)
        cout << "You qualify to run for President\n";
    else
        cout << "You are too young to run for President\n";
else
    cout << "You must have been born in the US in order "
        << "to run for President" << endl;
```

18

## Testing a series of conditions

- Decision structure to determine a grade



19

## Common nested if pattern

- Determine letter grade from test score:

```
if (testScore >= 90)
    grade = 'A';
else {
    if (testScore >= 80)
        grade = 'B';
    else {
        if (testScore >= 70)
            grade = 'C';
        else {
            if (testScore >= 60)
                grade = 'D';
            else
                grade = 'F';
        }
    }
}
```

If we are in this else branch, what do we know about the value of testScore?

- Note the braces are actually optional here!

## 4.6 The if/else if Statement

- Not really a different statement, just a different way of indenting the nested if statement from the previous slide:

```
if (testScore >= 90)
    grade = 'A';
else if (testScore >= 80)
    grade = 'B';
else if (testScore >= 70)
    grade = 'C';
else if (testScore >= 60)
    grade = 'D';
else
    grade = 'F';
```

- removed braces, put “if (...)” on previous line
- eliminated nested indentation.

21

## 4.8 Logical Operators

- Used to create relational expressions from other relational expressions:
  - ▶ **&&** AND (binary operator)  
**a && b** is true only when both a and b are true
  - ▶ **||** OR (binary operator)  
**a || b** is true whenever either a or b is true
  - ▶ **!** NOT (unary operator)  
**!a** is true when a is false

22

## Logical Operators

- Examples

```
int x=6;
int y=10;
```

- a. `x == 5 && y <= 3`
- b. `x > 0 && x < 10`
- c. `x == 10 || y == 10`
- d. `x == 10 || x == 11`
- e. `!(x > 0)`
- f. `!(x > 6 || y == 10)`

false && false is false  
true && true is true  
false || true is true  
\_\_\_\_ || \_\_\_\_ is \_\_\_\_  
!true is \_\_\_\_  
!( false || true) is \_\_\_\_

```
bool flag;
flag = (x > 0 && x < 25);
g. !flag
h. flag || x < 100
```

23

## Logical Operator Precedence

- **!** is higher than most operators, so use parentheses:

```
int x;
... !(x < 0 && x > -10) ... // <, >, &&, !
```

- **&&** is higher than **||**

```
int x, y;
bool flag;
... flag || x * 5 >= y + 10 && x == 5
// which op is first? second? etc?
```

- **&&** and **||** are lower than arithmetic+relational operators: parens not usually needed

24

## 4.9 Checking Numeric Ranges

- We want to know if x is in the range from 1 to 10 (inclusive)

a. 

```
if (1 <= x <= 10) //as in math class
    cout << "YES" << endl;

//THIS DOES NOT WORK IN C++:
// ((1<=x) <=10) (assume x is -5)
// => ( false <= 10)
// => ( 0<=10 ) is true, but should be false
```

b. 

```
if (1 <= x && x <= 10)
    cout << "YES" << endl;
```

-check: x=0? (1<=0 && 0<=10) => false && true  
-check: x=5? (1<=5 && 5<=10) => true && true  
-check: x=100? (1<=100 && 100<=10) => ?? <sup>25</sup>

## 4.10 Menus

- Menu-driven program: program controlled by user selecting from a list of actions
- Menu: list of choices on the screen
- Display list of numbered/lettered choices
- Prompt user to make a selection
- Test the selection in nested if/else or switch
  - ▶ Match found: execute corresponding code
  - ▶ Else: error message (invalid selection).

26

## Sample menu code

```
int choice;
double charges;
int months = 12;

// Display the menu and get a 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 << "Enter your choice: ";
cin >> choice;

// Respond to the user's menu selection.
if (choice==1) {
    charges = months * 40.0;
    cout << "The total charges are $" << charges << endl;
} else if (choice==2) {
    charges = months * 20.0;
    cout << "The total charges are $" << charges << endl;
} else if (choice==3) {
    charges = months * 30.0;
    cout << "The total charges are $" << charges << endl;
} else {
    cout << "ERROR: The valid choices are 1 through 3." << endl;
}
```

27

## 4.11 Validating User Input

- Input validation: inspecting input data to determine whether it is acceptable
- Invalid input is an error that should be treated as an exceptional case.
  - ▶ The program can ask the user to re-enter the data
  - ▶ The program can exit with an error message

```
cout << "Enter a positive number: ";
cin >> x;
if (x > 0) {
    //do something with x here
} else {
    cout << "You entered a negative number or 0." << endl;
    cout << "The program is ending." << endl;
}
```

28

## 4.12 Comparing Characters and Strings

- Characters are compared using their ASCII values

```
'A' < 'B'
```

- ▶ This is true.  
ASCII value of 'A' (65) is less than the ASCII value of 'B' (66)

```
'1' < '2'
```

- ▶ This is true.  
ASCII value of '1' (49) is less than the ASCII value of '2' (50)

- Lowercase letters have higher ASCII codes than uppercase letters, so 'a' > 'Z'

29

## Comparing string objects

- Like characters, strings are compared using their ASCII values

```
string name1 = "Mary";  
string name2 = "Mark";  
  
name1 > name2    // true  
name1 <= name2   // false  
name1 != name2   // true  
  
name1 < "Mary Jane" // true
```

The characters in each string must match exactly in order to be equal

Otherwise, use first non-equal character as basis of the comparison ('y' > 'k')

If a string is a prefix of the other, then it is less than the other

30

## 4.14 The switch statement

- Like a nested if/else, used to select one of multiple alternative code sections.
- tests **one** integer/char expression against **multiple constant** integer/char values:

```
switch (expression) {  
    case const1: statements  
    ...  
    case constn: statements  
    default: statements  
}
```

31

## switch statement behavior

```
switch (expression) {  
    case const1: statements  
    ...  
    case constn: statements  
    default: statements  
}
```

- expression is evaluated to an int/char value
- execution starts at the case labeled with that int/char value
- execution starts at default if the int/char value matches none of the case labels

32



## switch statement syntax

```
switch (expression) {  
    case const1: statements  
    ...  
    case constn: statements  
    default: statements  
}
```

- expression must have int/char type
- const1, constn must be constants!  
a literal or named constant
- statements is one or more statements  
(braces not needed and not recommended!)
- default: is optional

33

## switch statement example

- Example:

```
int quarter;  
...  
switch (quarter) {  
    case 1: cout << "First";  
            break;  
    case 2: cout << "Second";  
            break;  
    case 3: cout << "Third";  
            break;  
    case 4: cout << "Fourth";  
            break;  
    default: cout << "Invalid choice";  
}
```

34

## The break Statement

- The break statement causes an immediate exit from the switch statement.
- Without a break statement, execution continues on to the next set of statements (the next case).
- Sometimes this is useful: the textbook has some nice examples.

35

## Multiple labels

- if ch is 'a', it falls through to output "Option A" (then it breaks)

```
char ch;  
...  
switch (ch) {  
    case 'a':  
    case 'A': cout << "Option A";  
            break;  
    case 'b':  
    case 'B': cout << "Option B";  
            break;  
    case 'c':  
    case 'C': cout << "Option C";  
            break;  
    default: cout << "Invalid choice";  
}
```

36

## 4.15 More about blocks and scope

- The scope of a variable is the part of the program where the variable may be used.
- The scope of a variable is the innermost block in which it is defined, from the point of definition to the end of that block.
- Note: the body of the main function is just one big block.

37

## Scope of variables in blocks

```
int main()
{
    double income; //scope of income is red + blue
    cout << "What is your annual income? ";
    cin >> income;

    if (income >= 35000) {
        int years; //scope of years is blue;
        cout << "How many years at current job? ";
        cin >> years;
        if (years > 5)
            cout << "You qualify.\n";
        else
            cout << "You do not qualify.\n";
    }
    else
        cout << "You do not qualify.\n";
    cout << "Thanks for applying.\n";
    return 0;
}
```

Cannot access years down here

38

## Variables with the same name

- In an inner block, a variable is allowed to have the same name as a variable in the outer block.
- When in the inner block, the outer variable is not available (it is hidden).
- Not good style: difficult to trace code and find bugs
- See example next slide

39

## Variables with the same name

```
int main()
{
    int number;
    cout << "Enter a number greater than 0: ";
    cin >> number;
    if (number > 0) {
        int number; // another variable named number
        cout << "Now enter another number ";
        cin >> number;
        cout << "The second number you entered was ";
        cout << number << endl;
    }
    cout << "Your first number was " << number << endl;
}
```

```
Enter a number greater than 0: 88
Now enter another number 2
The second number you entered was 2
Your first number was 88
```

40