

Week 4: If statements and boolean expressions

Gaddis: 4.1-4.9

CS 1428
Fall 2015

Jill Seaman

1

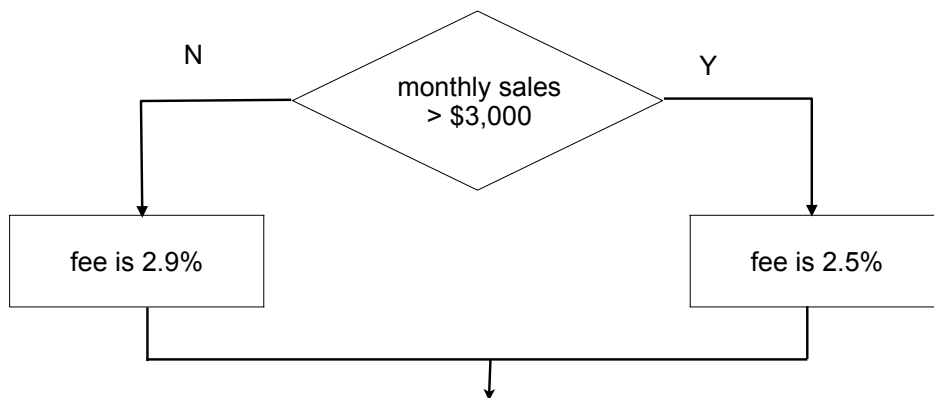
Straight-line code

- 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

- 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	_____
d. 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 evaluates to _____

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

4.4 if-else statement

- if-else statement is used to make decisions

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

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).

9

if-else example

```
double rate;
double monthlySales;

cout << "Enter monthly sales last month: " ;
cin >> monthlySales;

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

double price;
cout << "Enter selling price of item: " ;
cin >> price;
double commission = (price + 3.99) * rate;
cout << "Commission: $" << commission << endl;
```

```
Enter monthly sales last month: 3025
Enter selling price of item: 100
Commission: $2.59975
```

10

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!!
- The semi-colons belong to the statements,
not to the if-else

11

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.

12

if-else with blocks

- We can use blocks to put more than one statement in the branches of the if-else:

```
int number;
cout << "Enter a number" << endl;
cin >> number;

if (number % 2 == 0)
{
    number = number / 2;
    cout << "Even";
}
else
{
    number = (number - 1) / 2;
    cout << "Odd";
}
```

13

4.2 if statement

- The else part of the if-else stmt is optional:

```
if (expression)
    statement
```

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

14

if statement example

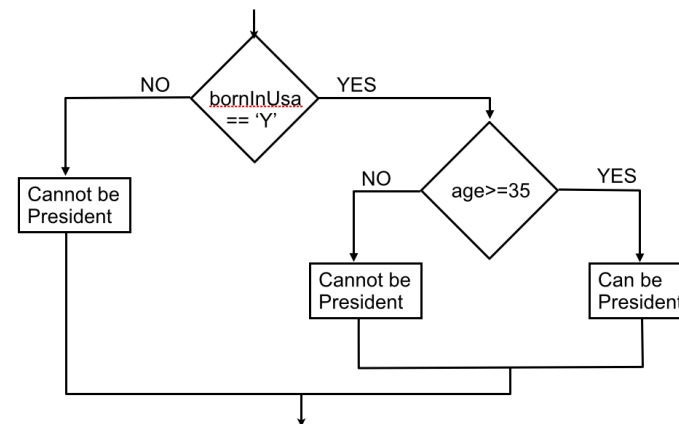
- Example: input validation

```
cout << "Enter a positive number: ";
cin >> x;
if (x <= 0)
{
    cout << "That number is not positive. "
        << "Please enter a positive number: ";
    cin >> x;
}
//do something with x here
```

15 Q1,2,3,4,5

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

Common nested if pattern

- Determine letter grade from test score:

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

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

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

20

Q6

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

21

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
```

22

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

23

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;
```

```
//WRONG: ((1<=x) <=10) (assume x is -5)
//      => ( false <= 10)
//      => ( 0 <= 10 ) is true
```

```
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) => ??

24