

Chapter 5 Topics

- **Data Type `bool`**
- **Using Relational and Logical Operators to Construct and Evaluate Logical Expressions**
- ***If-Then-Else* Statements**
- ***If-Then* Statements**
- **Nested If Statements for Multi-way Branching**

1

Flow of Control

- the order in which program statements are executed

WHAT ARE THE POSSIBILITIES...

2

Flow of Control

- is **Sequential** unless a “control structure” is used to change that
- there are 2 general types of control structures:
 - Selection** (also called branching)
 - Repetition** (also called looping)

3

bool Data Type

- type `bool` is a built-in type consisting of just 2 values, the constants `true` and `false`
- we can declare variables of type `bool`

```
bool hasFever; // true if has high temperature
bool isSenior; // true if age is at least 55
```

4

6 Relational Operators

are used in expressions of form:

<i>ExpressionA</i>	<i>Operator</i>	<i>ExpressionB</i>
--------------------	-----------------	--------------------

temperature > humidity

B * B - 4.0 * A * C > 0.0

abs (number) == 35

initial != 'Q'

6

```
int x, y ;
x = 4;
y = 6;
```

<u>EXPRESSION</u>	<u>VALUE</u>
x < y	true
x + 2 < y	false
x != y	true
x + 3 >= y	true
y == x	false
y == x+2	true
y = x + 3	7 (true)

7

LOGICAL		
EXPRESSION	MEANING	DESCRIPTION
! p	NOT p	! p is false if p is true ! p is true if p is false
p && q	p AND q	p && q is true if both p and q are true. It is false otherwise.
p q	p OR q	p q is true if either p or q or both are true. It is false otherwise.

10

```

int age ;
bool isSenior, hasFever ;
float temperature ;

age = 20;
temperature = 102.0 ;
isSenior = (age >= 55) ;           // isSenior is false
hasFever = (temperature > 98.6) ; // hasFever is true
    
```

<u>EXPRESSION</u>	<u>VALUE</u>
isSenior && hasFever	false
isSenior hasFever	true
! isSenior	true
! hasFever	false

11

What is the value?

```
int age, height;
```

```
age = 25;
```

```
height = 70;
```

EXPRESSION	VALUE
!(age < 10)	?
!(height > 60)	?

12

Write an expression for each

taxRate is over 25% and income is less than \$20000

temperature is less than or equal to 75 or humidity is less than 70%

age is over 21 and age is less than 60

age is 21 or 22

18

Some Answers

(taxRate > .25) && (income < 20000)

(temperature <= 75) || (humidity < .70)

(age > 21) && (age < 60)

(age == 21) || (age == 22)

19

Selection statements

**are used to choose an action
depending on the current situation
in your program as it is running**

21

Control structures

use logical expressions which may include:

6 Relational Operators

< <= > >= == !=

3 Logical Operators

! && ||

22

If-Then-Else Syntax

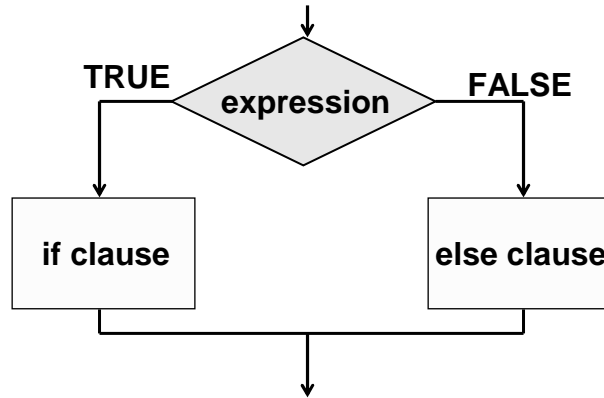
```
if ( Expression )  
    StatementA  
else  
    StatementB
```

NOTE: StatementA and StatementB each can be a single statement, a null statement, or a block.

25

if ... else provides two-way selection

between executing one of 2 clauses (the if clause or the else clause)



26

Use of blocks recommended

```
if ( Expression )
```

```
{
```

```
}
```

```
else
```

```
{
```

```
}
```

> "if clause"

> "else clause"

27

Multi-alternative Selection

is also called multi-way branching, and can be accomplished by using NESTED if statements.

30

Nested if Statements

```
if ( Expression1 )  
    Statement1  
else if ( Expression2 )  
    Statement2  
    .  
    .  
else if ( ExpressionN )  
    StatementN  
else  
    Statement N+1
```

EXACTLY 1 of these statements will be executed. ³¹

Nested if Statements

Each Expression is evaluated in sequence, until some Expression is found that is true.

Only the specific Statement following that particular true Expression is executed.

If no Expression is true, the Statement following the final else is executed.

Actually, the final else and final Statement are optional. If omitted, and no Expression is true, then no Statement is executed.

AN EXAMPLE . . .

32

Multi-way Branching

```
if ( creditsEarned >= 90 )
    cout << "SENIOR STATUS ";
else if ( creditsEarned >= 60 )
    cout << "JUNIOR STATUS ";
else if ( creditsEarned >= 30 )
    cout << "SOPHOMORE STATUS ";
else
    cout << "FRESHMAN STATUS ";
```

33

Switch Statement

Is a selection control structure for multi-way branching.

SYNTAX

```
switch ( IntegralExpression )
{
    case Constant1 :
        Statement(s);           // optional
    case Constant2 :
        Statement(s);           // optional
        .
        .
        .
    default :
        Statement(s);           // optional
}
```

40

```
float weightInPounds = 165.8 ;
char weightUnit ;
. . . // user enters letter for desired weightUnit
switch ( weightUnit )
{
    case 'P' :
    case 'p' :
        cout << weightInPounds << " pounds " << endl ;
        break ;
    case 'O' :
    case 'o' :
        cout << 16.0 * weightInPounds << " ounces " << endl ;
        break ;
    case 'K' :
    case 'k' :
        cout << weightInPounds / 2.2 << " kilos " << endl ;
        break ;
    case 'G' :
    case 'g' :
        cout << 454.0 * weightInPounds << " grams " << endl ;
        break ;
    default :
        cout << "That unit is not handled! " << endl ;
        break ;
}
```

41

Switch Statement

- the value of *IntegralExpression* (of char, short, int, long or enum type) determines which branch is executed
- case labels are constant (possibly named) integral expressions. Several case labels can precede a statement

42

Control in Switch Statement

- control branches to the statement following the case label that matches the value of *IntegralExpression*. Control proceeds through all remaining statements, including the default, unless redirected with `break`
- if no case label matches the value of *IntegralExpression*, control branches to the default label, if present--otherwise control passes to the statement following the entire switch statement
- forgetting to use `break` can cause logical errors because after a branch is taken, control proceeds sequentially until either `break` or the end of the switch statement occurs

43