# Selection Statements or Conditionals

ITSC 2181: Introduction to Computer Systems
UNC Charlotte
College of Computing and Informatics

COLLEGE OF COMPUTING AND INFORMATICS

# Flow of Control

- Flow-of-control statements in C:
  - **`if-then-else`**
  - **`conditional operator (? : )`**
  - **`switch-case`**
  - **`for`**
  - **`continue and break`**
  - **`while`** and **`do-while`**

## The **`if`** statement

- Allows a program to choose between two alternatives by evaluating an expression.
- Syntax:

```
if (expression) statement
```

- Example:

```
if (grade > 95)
   printf("A+");
```

COLLEGE OF COMPUTING AND INFORMATICS

## Relational and Logical Operators

Used in evaluation conditions

```
if (expression evaluates to TRUE)
    ...do something...
```

What is TRUE (in C)?

- **0** means FALSE
- **anything else (1, -96, 1.414,‘F’, inf)** means TRUE
- ???

```
float f = 9593.264;
if (f)
    ...do something...
```

COLLEGE OF COMPUTING AND INFORMATICS

## Relational Operators

Six comparison operators: `<, >, ==, !=, >=, <=`

```
if (a < b) ...
if (x >= y) ...
if (q == r) ...
```

- Operands must be numbers (integer or floating point), result type is `int`
  - i.e., cannot use to compare structs, functions, arrays, etc.
- If relation is true, result is `1`, else result is `0`

```
float f = 9593.264;
if (f != 0)
    ...do something...
```

same meaning
as in previous slide

COLLEGE OF COMPUTING AND INFORMATICS

# Relational Operators

- C's ***relational operators:***

|  |  |
|---|---|
| `<` | less than |
| `>` | greater than |
| `<=` | less than or equal to |
| `>=` | greater than or equal to |

(see `if_stmts.c` in *Code samples and Demonstrations* in *Canvas*)

- Produce `0` (false) or `1` (true) when used in expressions.
- Can be used to compare integers and floating-point numbers, with operands of mixed types allowed.

COLLEGE OF COMPUTING AND INFORMATICS

## Relational Operators (cont'd)

- **One of the most common mistakes in C**
  - **==  is relational comparison for equality**
  - = is assignment!

☠ *common source of bugs* ☠
**confusion between**
**= and ==**

Example: some strategic defense code…

```
if (enemy_launch = confirmed)
     retaliate();
```

Oops… sorry!

COLLEGE OF COMPUTING AND INFORMATICS

# Logical Operators

Logical operators allow construction of complex (compound) conditions

Operands must be (or return) numbers (integer or floating point), result type is `int`

Logical NOT (`!`) operator

– result: `1` (TRUE) if operand was `0` (FALSE), otherwise `0`

```
int j = ...;
if (! j)
    ... do something ...
```

```
float f = ..., g = ...;
if (! (f < g) )
    ... do something ...
```

COLLEGE OF COMPUTING AND INFORMATICS

## Logical … (cont'd)

- AND (`&&`):
  - evaluate **first** operand, if **0**, result is **0**; else,
  - evaluate **second** operand, if **0**, result is **0**; else,
  - result is **1**

```
if (x && (y > 32))
    ... do something ...
```

COLLEGE OF COMPUTING AND INFORMATICS

# Logical… (cont'd)

- Condition evaluation stops as soon as truth value is **known**, short-circuit evaluation
  - i.e., **order** of the operands is **significant**
- Relied on by many programs!

☠ *common source of bugs* ☠
**lack of understanding of significance of order in conditions**

```
if ((b != 0) && ((a / b) > 5))
     printf("quotient greater than 5\n");
```

what's the difference???

```
if (((a / b) > 5) && (b != 0))
     printf("quotient greater than 5\n");
```

COLLEGE OF COMPUTING AND INFORMATICS

## Logical… (cont'd)

- OR ( `||` ) operator
  - evaluate **first** operand, if **not 0**, result is **1**;
  - otherwise, evaluate **second** operand, if **not 0**, result is **1**;
  - otherwise, result is **0**
- There is **no logical XOR** in C

  (**a** XOR **b**) → `(a && (! b)) || ((!a) && b)`

COLLEGE OF COMPUTING AND INFORMATICS

# The **`else`** clause

- **`if`** statements can have an else clause.
- The statement that follows **`else`** is executed if the expression evaluates to zero (*false*).
- Syntax:

```
if (expression) statement
else statement
```

## The **else** clause: Example

```
if (age > 16)
   printf("Can drive");
else
   printf("Too young to drive");
```

(see **if_then_else.c** in *Code samples and Demonstrations* in *Canvas*)

COLLEGE OF COMPUTING AND INFORMATICS

# Using Compound Statements

- Any group of statements that is surrounded by braces will be handled by the C compiler as a single statement.
- Syntax:

```
{
    statement1;
    statement2;
    …
    statementn;
}
```

## Compound Statement Example

```
if (age > 16)
   printf(“Can drive”);
else
{
   printf(“Too young to drive”);
   printf(“Please re-apply later”);
}
```

COLLEGE OF COMPUTING AND INFORMATICS

## Cascaded `if` statements

```
if (expression)
   statement
else if (expression)
   statement
…
else if (expression)
   statement
else
   statement
```

(see **broker.c** in *Code samples and Demonstrations* in *Canvas*)

COLLEGE OF COMPUTING AND INFORMATICS

# Cascaded `if` Statements

- A "cascaded" `if` statement is often the best way to test a series of conditions, stopping as soon as one of them is true.
- Example:

```
if (n < 0)
  printf("n is less than 0\n");
else
  if (n == 0)
    printf("n is equal to 0\n");
  else
    printf("n is greater than 0\n");
```

(see `broker.c` in *Code samples and Demonstrations* in *Canvas*)

COLLEGE OF COMPUTING AND INFORMATICS

# Cascaded `if` Statements

- Although the second `if` statement is nested inside the first, C programmers don't usually indent it.
- Instead, they align each `else` with the original `if`:

```
if (n < 0)
  printf("n is less than 0\n");
else if (n == 0)
  printf("n is equal to 0\n");
else
  printf("n is greater than 0\n");
```

## References

- S. J. Matthews, T. Newhall and K. C. Webb, *Dive into Systems*, Version 1.2. Free online textbook, available at: [https://diveintosystems.org/book/](https://diveintosystems.org/book/)

- K. N. King, *C Programming: A Modern Approach*, 2nd Edition. W. W. Norton & Company. 2008.

- D.S. Malik, *C++ Programming: From Problem Analysis to Program Design*, Seventh Edition. Cengage Learning. 2014.

COLLEGE OF COMPUTING AND INFORMATICS