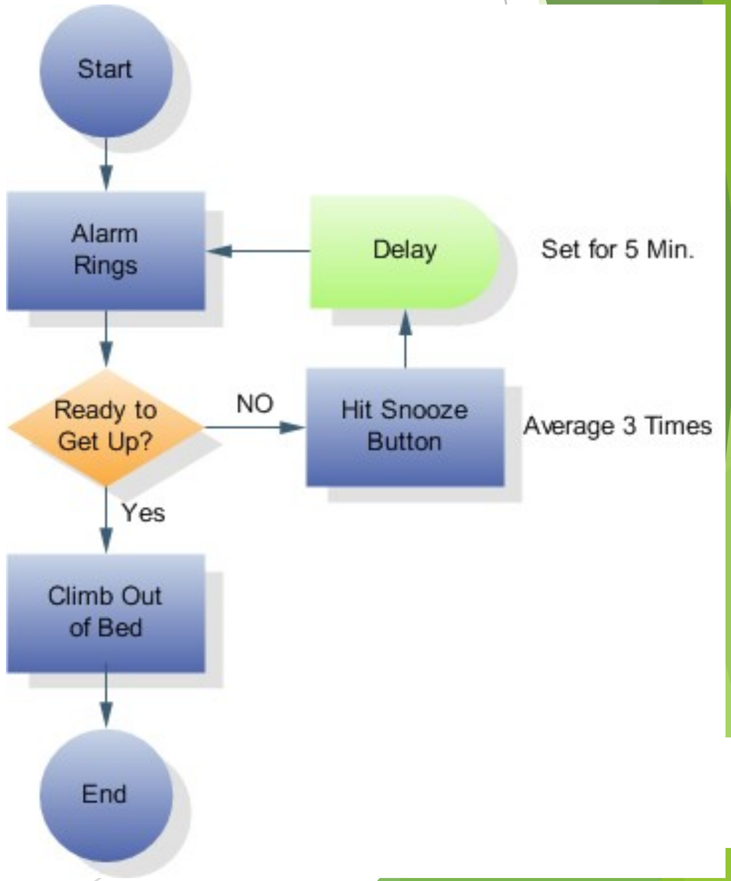
CSE103 Structured Programming Lecture-4

Dr. Maheen Islam
Associate Professor
Dept. of CSE
East West University

Flow of Control

- Flow of control
  - The order in which statements are executed
- Transfer of control
  - When the next statement executed is not the next one in sequence



Flow of Control

- Control structures

  combination of individual statements into a logical unit that regulates the flow of execution in a program or function

  - Sequence
  - Selection (Making Decisions)
  - Repetition (Looping)

Boolean Expressions

- Evaluate to true or false
- Forms
  - Relational expression: <expr> <relational operator> <expr>
    - Examples:

```
7 < 5
a + b > 6
```

  - Logical expression: <Boolean expr> <logical operator> <Boolean expr>
    - Examples:

```
(x < 7) && (y > 3)
```

Relational Operators

Standard Algebraic Relational Operator	C Relational Operator	C Condition Example	Meaning of C Condition
Inequality			
<	<	x < y	**x** is less than **y**
≤	<=	x <= y	**x** is less than or equal to **y**
>	>	x > y	**x** is greater than **y**
≥	>=	x >= y	**x** is greater than or equal to **y**
Equality			
=	==	x == y	**x** is equal to **y**
≠	!=	x != y	**x** is not equal to **y**

4th: Ch 4 p. 46
3rd: Ch 5 p. 46

Logical Operators (Compound Relationals)

- **`&&`** (logical **AND**)
  - Returns **`true`** if both conditions are **`true`**
- **`||`** (logical **OR**)
  - Returns **`true`** if either of its conditions is **`true`**
- **`!`** (logical **NOT**, logical negation)
  - Is a unary operator, only takes one operand following
  - Reverses the truth/falsity of its condition
  - Returns **`true`** when its condition is **`false`**

Logical Operators Truth Table

P	Q	P && Q	P | | Q	!P
true	true	true	true	false
true	false	false	true	false
false	true	false	true	true
false	false	false	false	true

Precedence of Operators

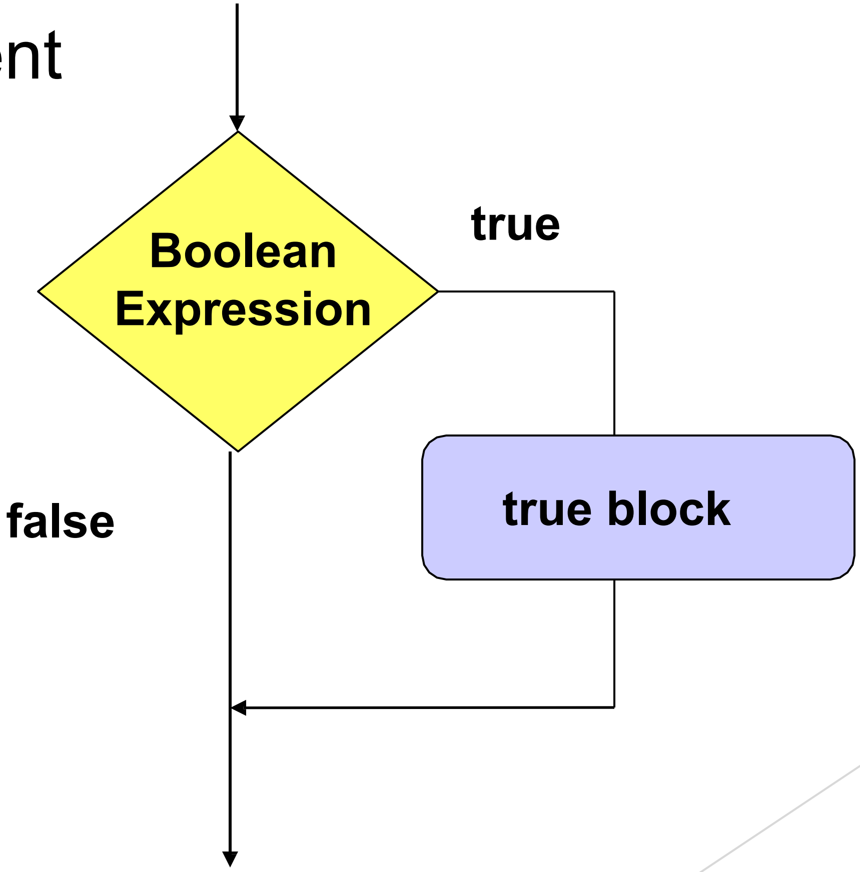
1. (), []
2. Unary +, unary -, !, ++,
--
3. Type casting
4. * , / , %
5. + , -
6. <, <=, >, >=
7. ==, !=
8. &&
9. ||
10. =

The `if` Selection Structure

- Selection structure
  - used when we want the computer to choose between two alternative courses of action

The `if` Selection Structure

- *if* Statement



The *if* Selection Structure

- General form of *if*:

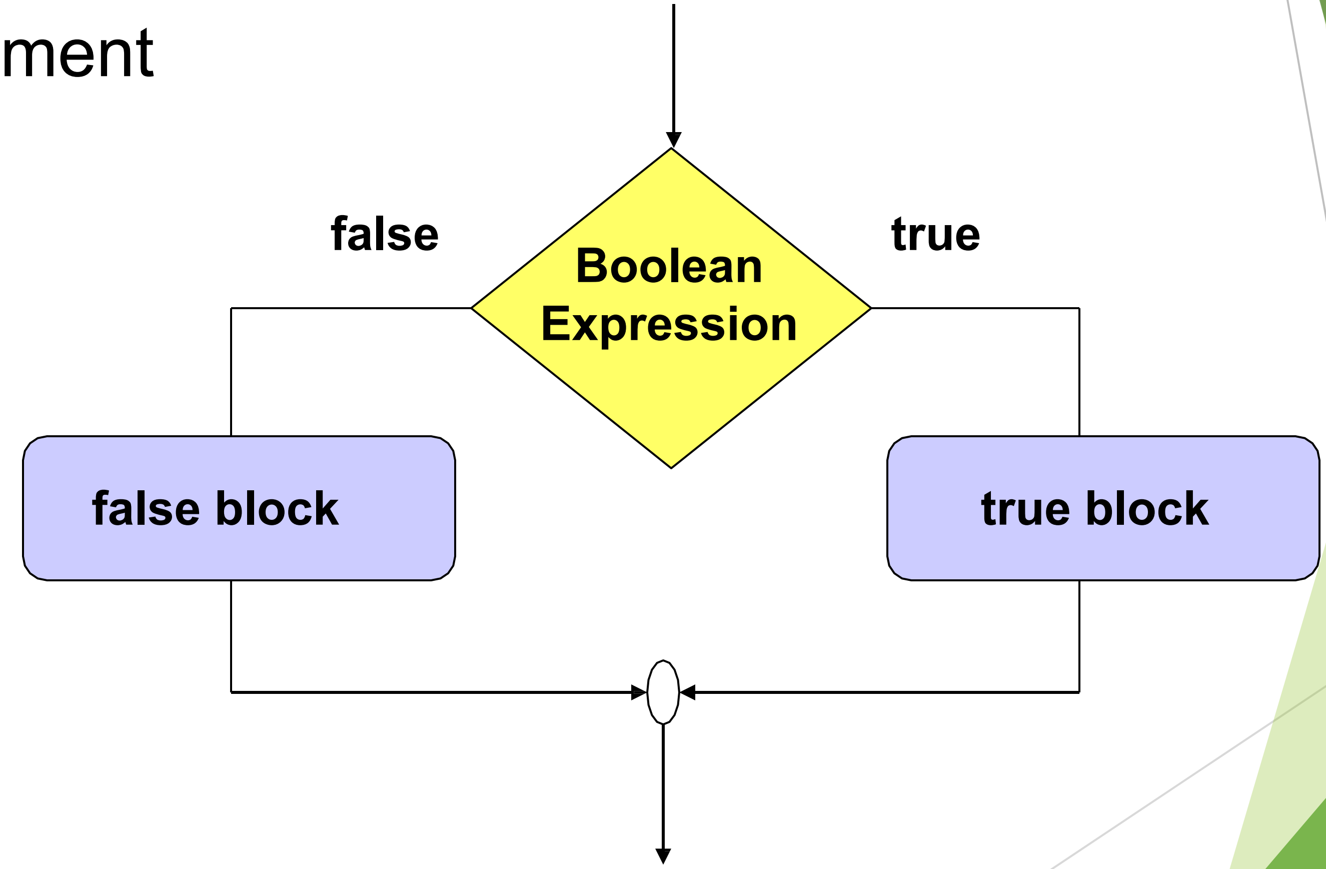
```
► if (Boolean Expression)
  ► {                   statement2;
            ►} statement1;
```

The `if-else` Selection Structure

- *`if`*
  - Only performs an action if the condition is true
- *`if-else`*
  - A different action is performed when condition is true and when condition is false

`if-else` Selection Structure

if-else
statement



The *if-else* Selection Structure

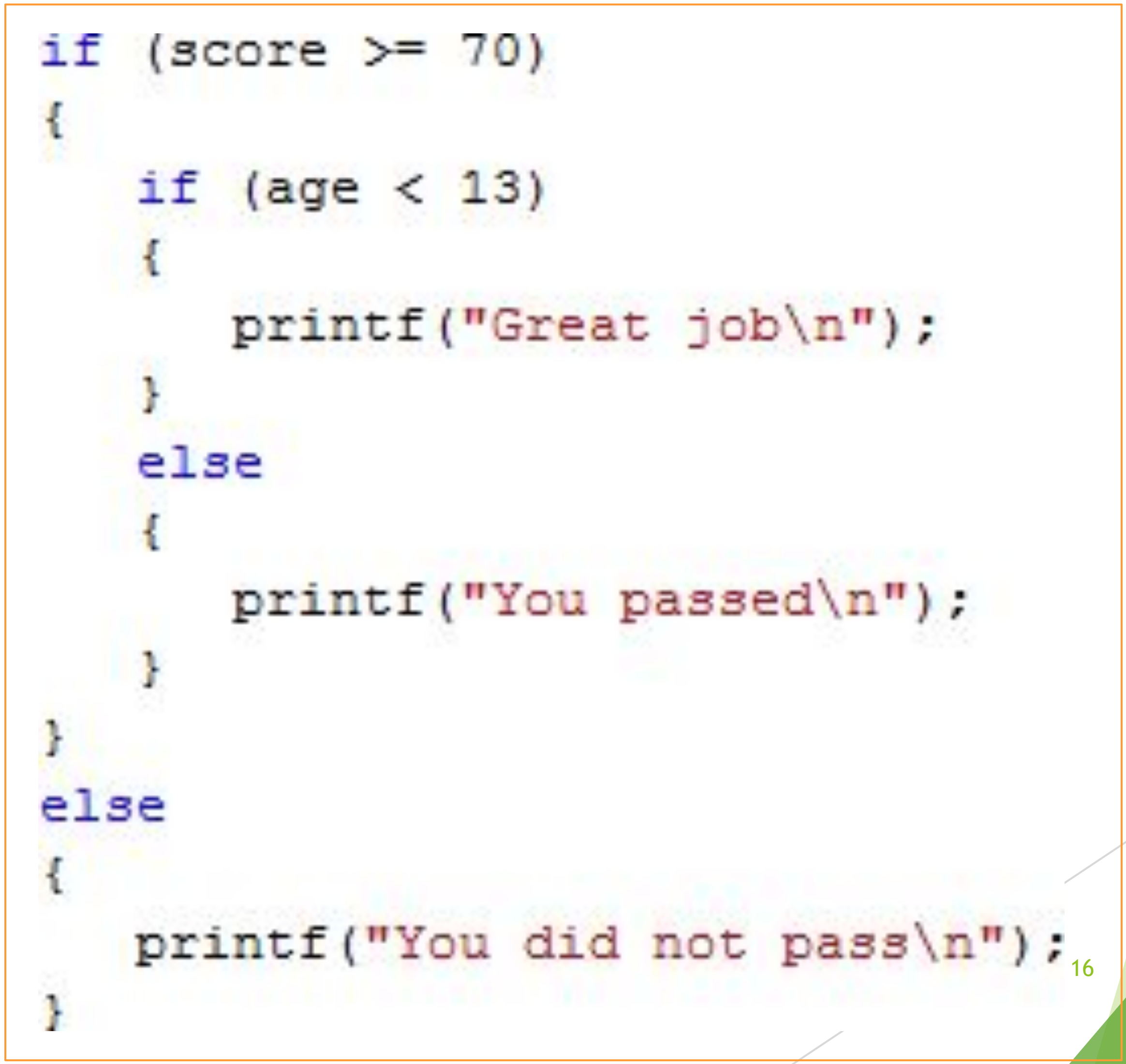
General form of *if-else*:

```
if (expression)
{
        statement1A;

        statement2A;

        ...
}
else
{
        statement1B;
        statement2B;
        ...
}
```

The `if-else` Selection Structure

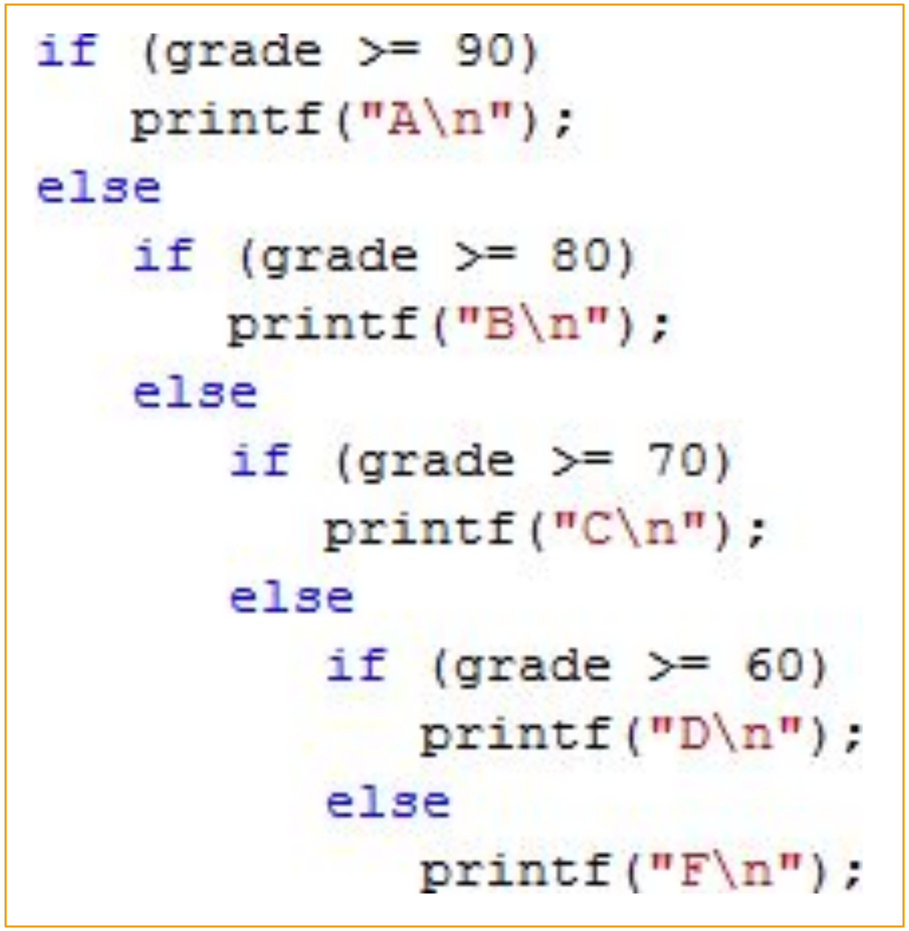
- Nested *`if-else`* structures
  - Test for multiple cases by placing **`if-else`** selection structures inside **`if-else`** selection structures.

Nested `if-else` Structures



```c
if (score >= 70)
{
    if (age < 13)
    {
        printf("Great job\n");
    }
    else
    {
        printf("You passed\n");
    }
}
else
{
    printf("You did not pass\n");
}
```

The `if-else-if` Construct

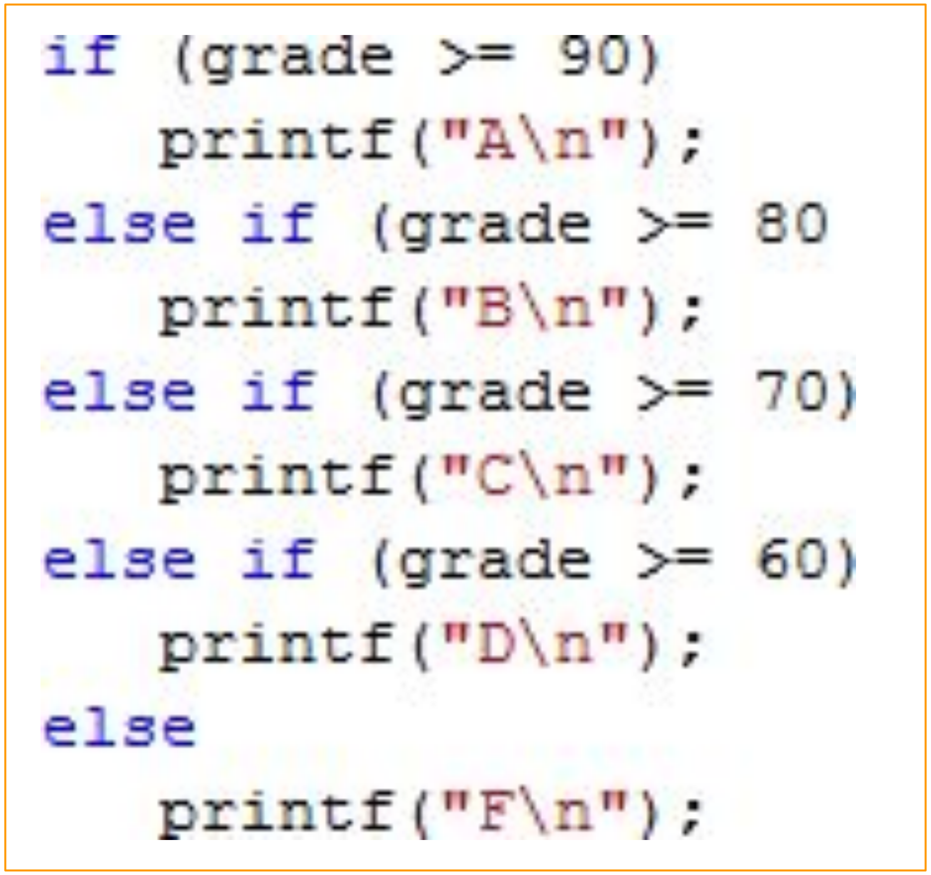


```
if (grade >= 90)
   printf("A\n");
else
   if (grade >= 80)
      printf("B\n");
   else
      if (grade >= 70)
         printf("C\n");
      else
         if (grade >= 60)
            printf("D\n");
         else
            printf("F\n");
```

- Once a condition is met, the rest of the statements are skipped

The *if-else-if* Construct

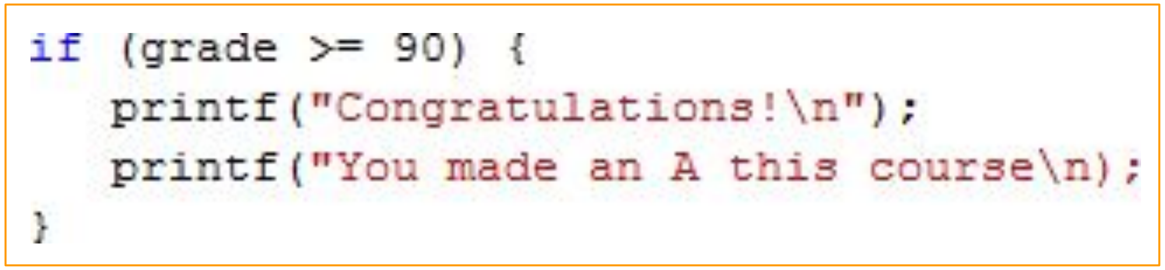
The standard way to indent the previous code is



```
if (grade >= 90)
   printf("A\n");
else if (grade >= 80
   printf("B\n");
else if (grade >= 70)
   printf("C\n");
else if (grade >= 60)
   printf("D\n");
else
   printf("F\n");
```

The *if-else* Selection Structure

- Compound statement:
  - Set of statements within a pair of braces
  - Example:



```c
if (grade >= 90) {
    printf("Congratulations!\n");
    printf("You made an A this course\n);
}
```

The *if-else* Selection Structure

–Without the braces, only one statement is executed.

e.g.   given the following code:

```
if (grade >= 90)
   printf("Congratulations!\n");
   printf("You made an A this course\n);
```

- The statement,

```
printf("You made an A this course\n);
```

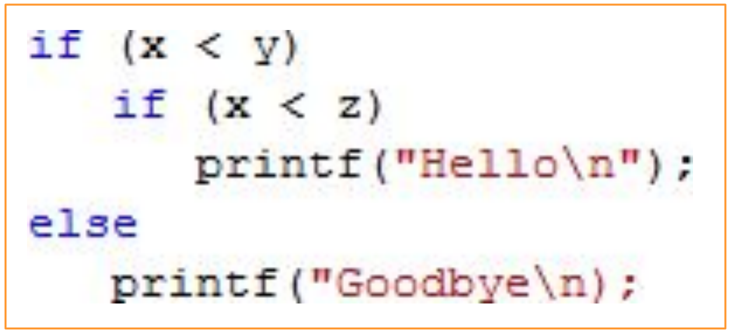
will be executed independent of the value of grade.

- The statement,

```
printf("Congratulations!\n");
```

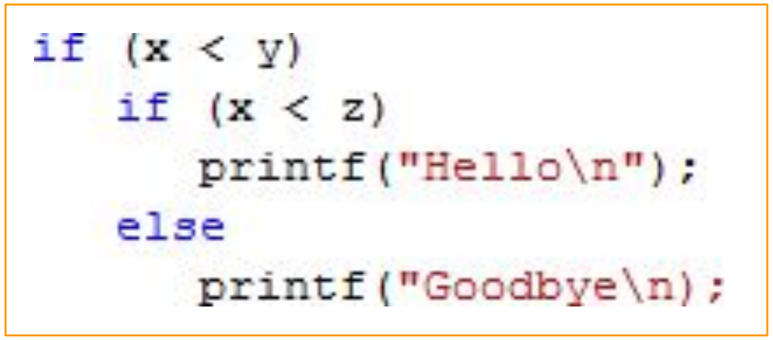
will execute only if grade is greater than or equal to 90.

The *dangling* else



```
if (x < y)
    if (x < z)
        printf("Hello\n");
else
    printf("Goodbye\n);
```

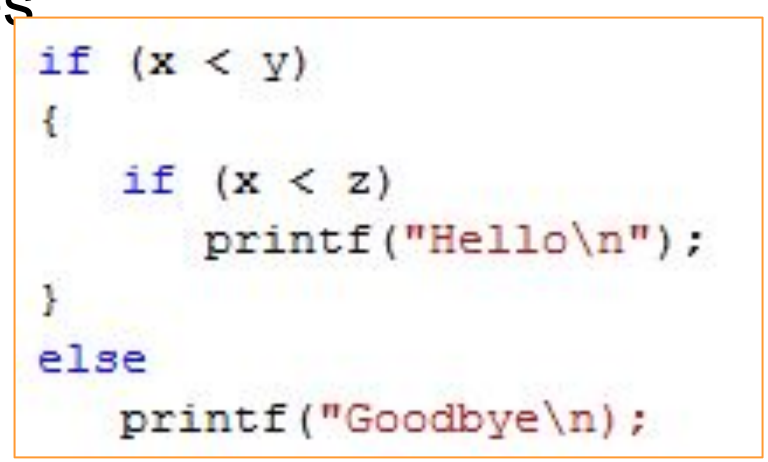
Note: the compiler matches an else with the closest unmatched if The above will be treated as



```
if (x < y)
    if (x < z)
        printf("Hello\n");
    else
        printf("Goodbye\n);
```

The *dangling* else

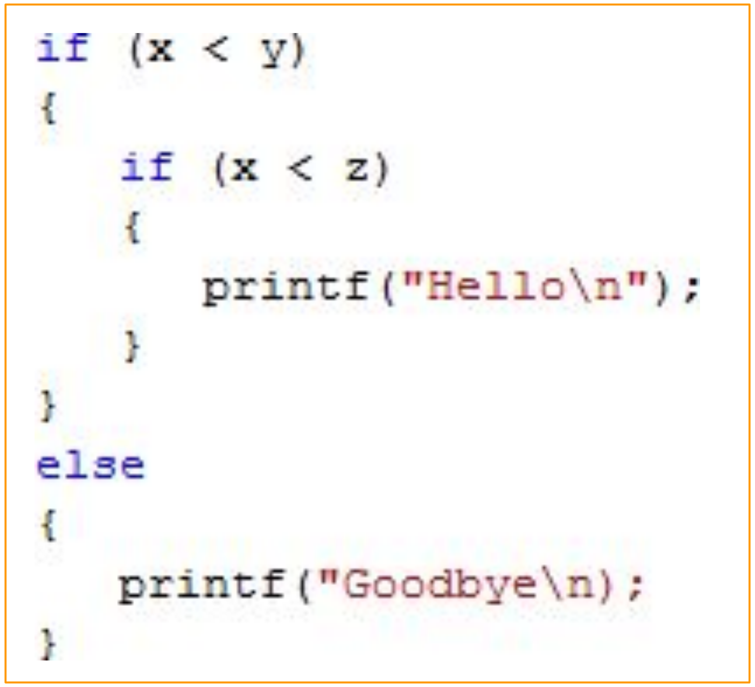
If the else is to match the outer if, use braces



```
if (x < y)
{
    if (x < z)
        printf("Hello\n");
}
else
    printf("Goodbye\n);
```

if-else Construct

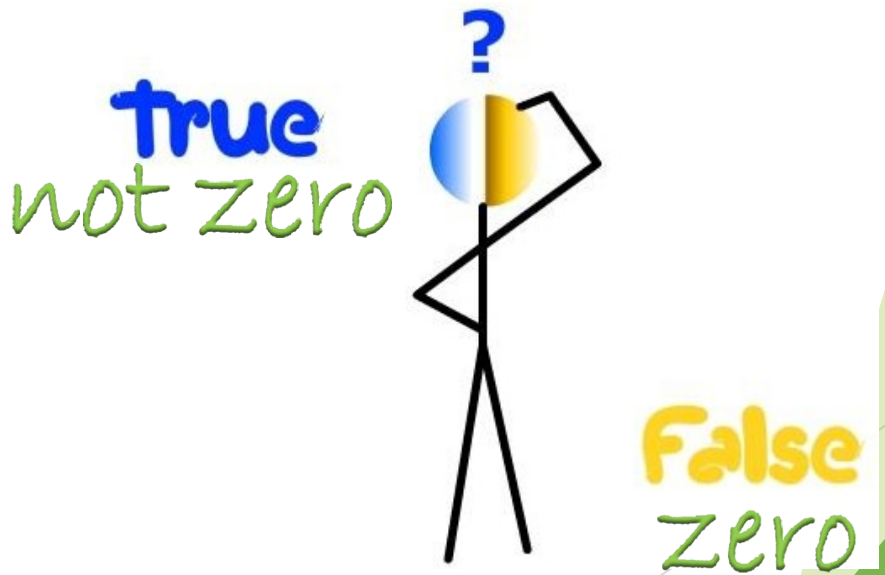
- To avoid confusion, and possible errors, it is best to use braces even for single statements.
  - However, code will be longer



```
if (x < y)
{
    if (x < z)
    {
        printf("Hello\n");
    }
}
else
{
    printf("Goodbye\n);
}
```

Conditionals

- C uses an integer to represent Boolean values
  - Zero is interpreted as false
  - Any other integer value is interpreted as true



Conditionals

- `if (n = 0)` is not a syntax error in C.
  - The expression, *n = 0*, assigns zero to n and the value of the expression is 0. Zero is interpreted as false, and the false branch of the if statement will be taken.
- `if (n = 5)` is not a syntax error in C.
  - The expression assigns 5 to n. 5 is interpreted as true, and the true branch of the if statement will be taken.

```
warning: suggest parentheses around assignment used as truth value
```

Conditionals

- Remember to use the == operator to test for equality.
- To help catch the error when the equality check involves a constant, put the constant on the left hand side of the ==

`if (0 == n)`

- For example, use

`if (n == 0)`

`0 = n` instead of

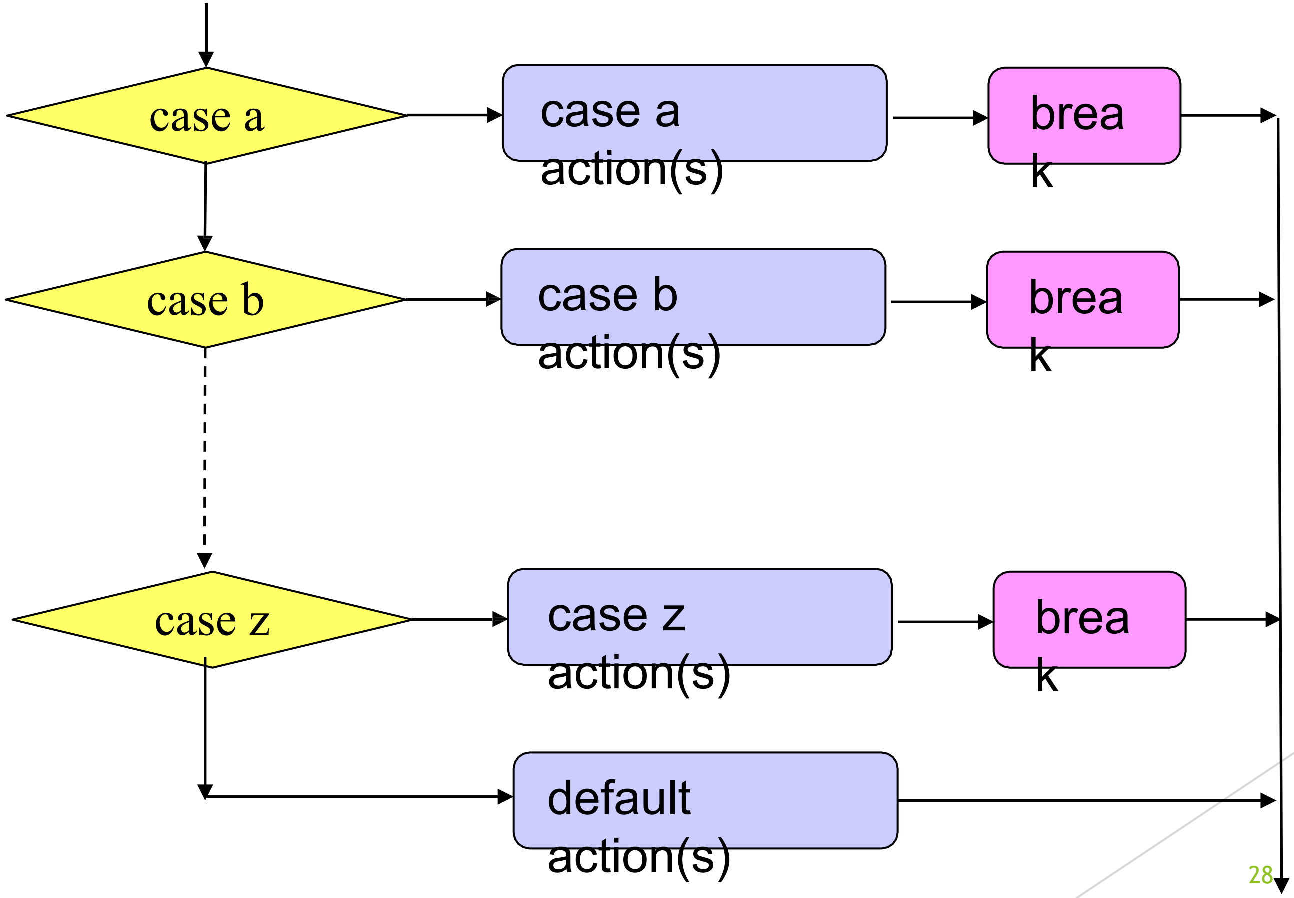
Since is not a valid assignment in C, the compiler will detect this error when == is intended.

```
error: invalid lvalue in assignment
```

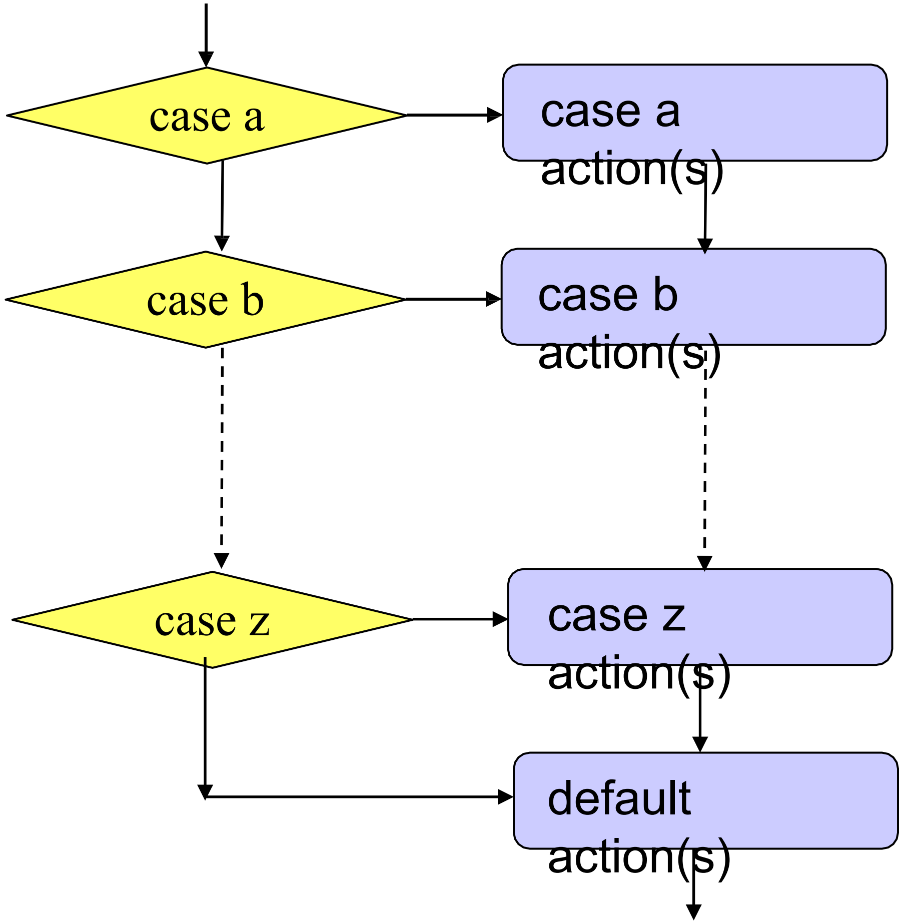
The `switch` Multiple-Selection Structure

- `switch`
  - Useful when variable or expression is tested for multiple values
  - Consists of a series of **`case`** labels and an optional **`default`** case

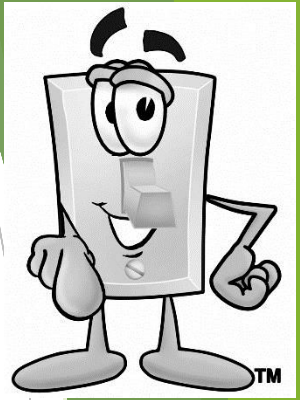
The *switch* Multiple-Selection Structure With Breaks



The `switch` Multiple-Selection Structure Without Breaks



`switch` Statement Syntax



```
switch (switch_expression)
{
        case constant1:
                statementSequence
                1  break;
        case constant2:
                statementSequence2
                break;
                              …
        case constantN:
                statementSequenceN
                break;
        default:
                defaultStmtSequence
}
```

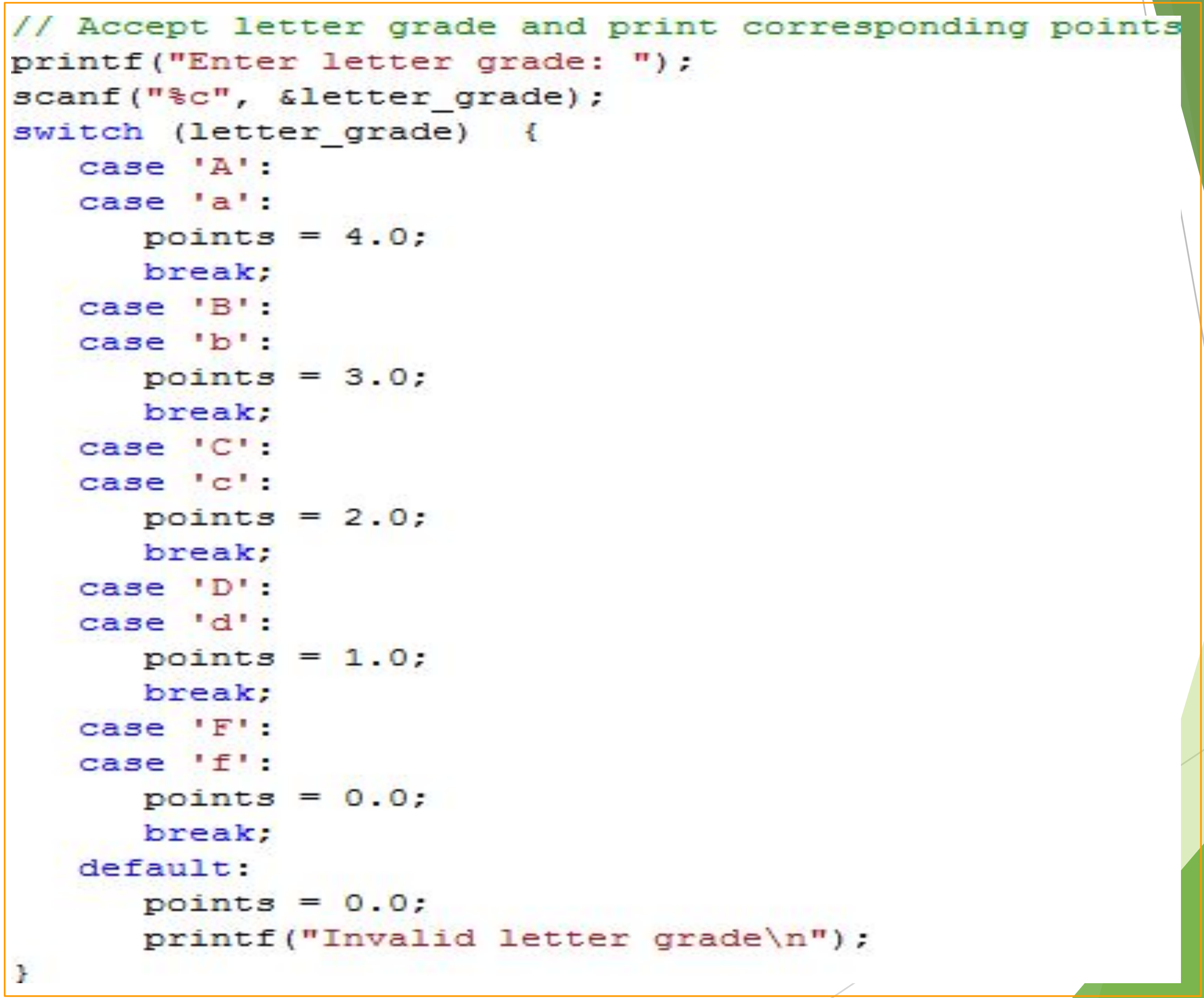
`switch` Statement

- The switch_expression is compared against the values
  constant1, constant2, …, constantN
  - *constant1, constant2, …, constantN* must be simple constants or constant expressions.
    - Can be a char or an int
    - Best to use the same type constant as the switch expression
      - If not, a type conversion will be done.

`switch` Statement Reminder

- The `switch` statement ends
  - break statement
  - end of the switch statement
- When executing the statements after a case label, it continues to execute until it reaches a break statement or the end of the switch.
- If you omit the break statements, then after executing the code for one case, the computer will continue to execute the code for the next case.

Example of *switch*



```
// Accept letter grade and print corresponding points
printf("Enter letter grade: ");
scanf("%c", &letter_grade);
switch (letter_grade)  {
    case 'A':
    case 'a':
        points = 4.0;
        break;
    case 'B':
    case 'b':
        points = 3.0;
        break;
    case 'C':
    case 'c':
        points = 2.0;
        break;
    case 'D':
    case 'd':
        points = 1.0;
        break;
    case 'F':
    case 'f':
        points = 0.0;
        break;
    default:
        points = 0.0;
        printf("Invalid letter grade\n");
}
```