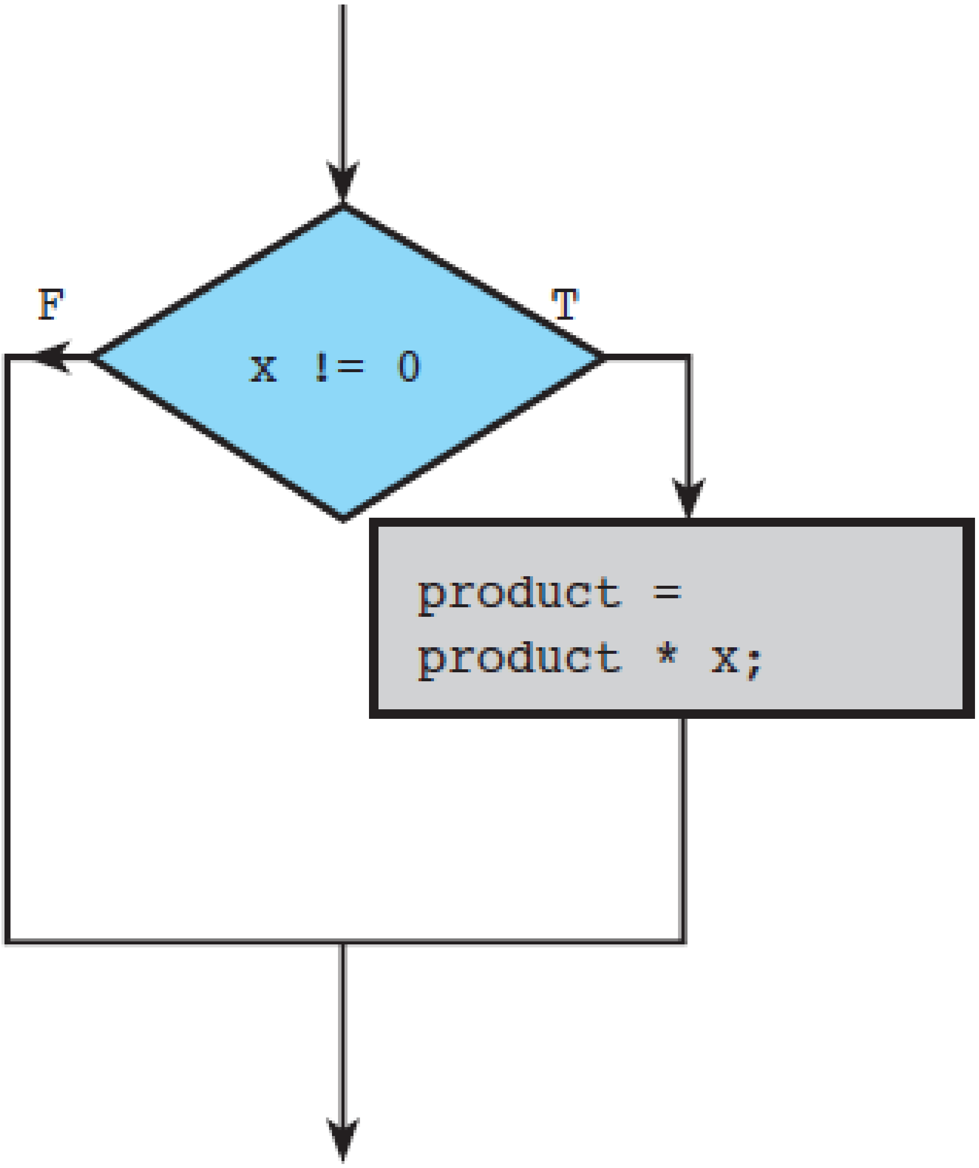
Lecture 06

Selection Structures

CSE115: Computing Concepts

The `if` Statement–1 Alternative

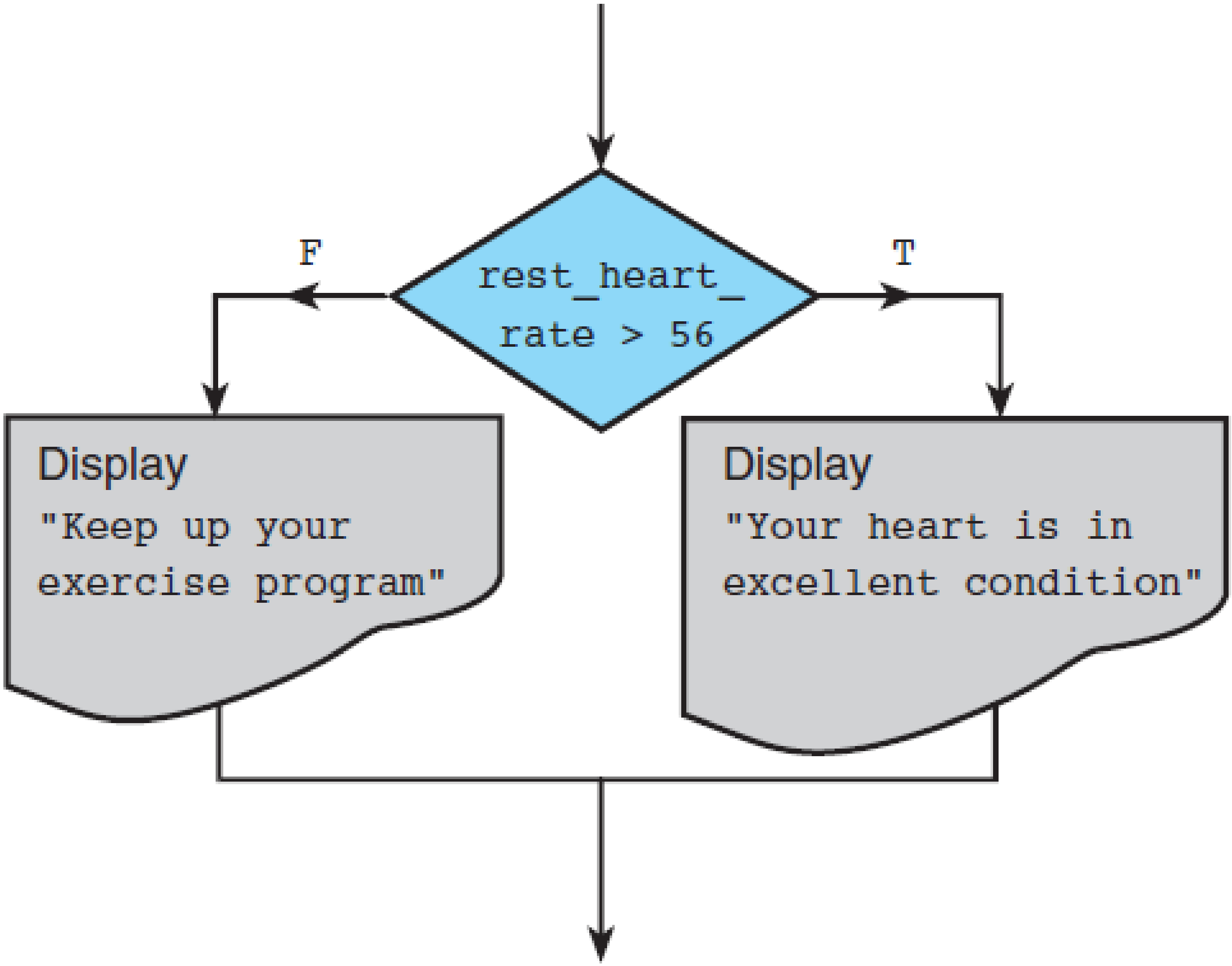


The **`if`** Statement – One Alternative

- **if Statement (One Alternative)**
  - FORM:
    ```
    if( condition )
          statement T ;
    ```
  - INTERPRETATION: If ***condition*** evaluates to **true** (a nonzero value), then ***statement T*** is executed; otherwise, *statement T* is skipped.
  - EXAMPLE:

```
if (x != 0)
   product = product * x;
```

The `if` Statement – Two Alternatives



The `if` Statement – 2 Alternatives

- **if Statement (Two Alternatives)**
  - FORM: `if( condition )`

```
          statement T ;
      else
          statement F ;
```

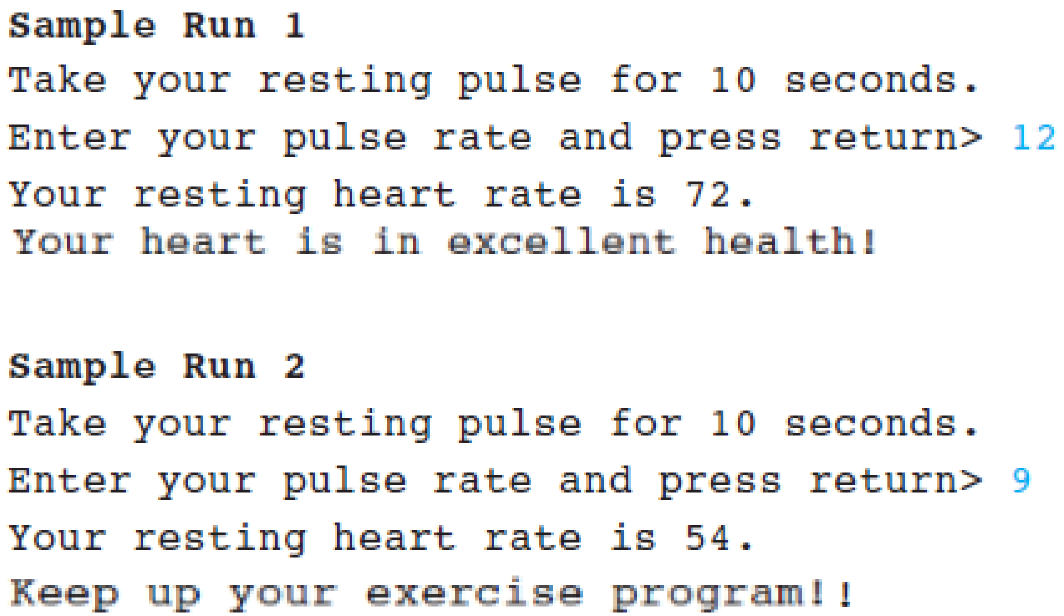
- INTERPRETATION: If ***condition*** evaluates to **true** (a nonzero value), then ***statement T*** is executed and *statement F* is skipped; otherwise, *statement T* is skipped and ***statement F*** is executed.
- EXAMPLE:

```
if (rest_heart_rate > 56)
    printf("Your heart is in excellent health!\n");
else
    printf("Keep up your exercise program!\n");
```

The `if` Statement – 2 Alternatives

```
#include <stdio.h>
int main()
{
    int pulse; /* resting pulse rate for 10 secs */
    int rest_heart_rate; /* resting heart rate for 1 minute */

    /* Enter your resting pulse rate */
    printf("Take your resting pulse for 10 seconds.\n");
    printf("Enter your pulse rate and press return> ");
    scanf("%d", &pulse);

    /* Calculate resting heart rate for minute */
    rest_heart_rate = pulse * 6;
    printf("Your resting heart rate is %d.\n", rest_heart_rate);

    /* Display message based on resting heart rate */
    if (rest_heart_rate > 56)
        printf("Your heart is in excellent health!\n");
    else
        printf("Keep up your exercise program!\n");

    return 0;
}
```

The `if` Statement – Two Alternatives



```
Sample Run 1
Take your resting pulse for 10 seconds.
Enter your pulse rate and press return> 12
Your resting heart rate is 72.
Your heart is in excellent health!

Sample Run 2
Take your resting pulse for 10 seconds.
Enter your pulse rate and press return> 9
Your resting heart rate is 54.
Keep up your exercise program!!
```

Look for Bugs!!!

- If the variable **`item`** is even, print “It’s an even number”, otherwise print “It’s an odd number”

```
if item % 2 == 1
    printf("It’s an odd number");
printf("It’s an even number");
```

```
if (item % 2 == 1);
   printf("It’s an odd number");
printf("It’s an even number");
```

```
if (item % 2 == 1)
   printf("It’s an odd number");
printf("It’s an even number");
```

```
if (item % 2 == 1)
     printf("It’s an odd number");
else
     printf("It’s an even number");
```

`if` Statements with Compound True or False Statements

- Enclose a compound statement that is a true task or a false task in braces.
- Placement of the braces is a matter of personal preference.

```
if ( condition )
{
   true task
}
else
{
   false task
}
```

`if` Statements with Compound True or False Statements

```
if (pop_today > pop_yesterday)
{
     growth = pop_today - pop_yesterday;
     growth_pct = 100.0 * growth / pop_yesterday;
     printf("The growth percentage is %.2f\n", growth_pct);
}
```

```
if (ctri <= MAX_SAFE_CTRI)
{
     printf("Car #%d: safe\n", auto_id);
     safe = safe + 1;
}
else
{
     printf("Car #%d: unsafe\n", auto_id);
     unsafe = unsafe + 1;
}
```

Indentation Style

- Acceptable

```
if (cond) {
  statements;
}
else {
  statements;
}
```

```
if (cond)
{
    statements;
}
else
{
    statements;
}
```

```
if (cond) {
  statements;
} else {
  statements;
}
```

- Not acceptable

```
if (cond)
{
statements;
}
else
{
statements;
}
```

No indentation!

```
if (cond) {
      statements; }
else {
      statements; }
```

Closing braces not aligned with if/else keyword.

Conditions

- Boolean values: true / false.
- Condition:
  - An expression that evaluates to a Boolean value (also called relational expression)
  - It is composed of expressions combined with relational or equality operators.
  - Examples:
  - `( a <= 10 ), ( count > max ), ( value != -9 )`
- There is no boolean type in ANSI C. Instead, we use integers:
  - 0 to represent false
  - Any other value to represent true (1 is used as the representative value for true in output)

Conditions

- Relational and Equality Operators

Relational Operator	Meaning	Type
<	is less than	relational
<=	is less than or equal to	relational
>	is greater than	relational
>=	is greater than or equal to	relational
==	is equal to	equality
!=	is not equal to	equality

Truth Values

- Example:

```
int a = (2 > 3);
int b = (3 > 2);

printf("a = %d; b = %d\n", a, b);
```

Truth Values

- Example:

```
int a = (2 > 3);
int b = (3 > 2);

printf("a = %d; b = %d\n", a, b);
```

```
a = 0; b = 1
```

Truth Values

- Be careful of the value returned/evaluated by a relational operation.
- Since the values 0 and 1 are the returned values for false and true respectively, we can have codes like these:

```
int a = 12 + (5 >= 2);           // 13 assigned to a
int b = (4 > 5) < (3 > 2) * 6;   // 1 assigned to b
int c = ( (4 > 5) < (3 > 2) ) * 6;// 6 assigned to c
```

- You are certainly not encouraged to write such convoluted codes!

Logical Operators

- Complex conditions: combine two or more boolean expressions.
- Examples:
  - If temperature is greater than 40C or blood pressure is greater than 200, go to hospital immediately.
  - If all the three subject scores (English, Maths and Science) are greater than 85 and mother tongue score is at least 80, recommend taking Higher Mother Tongue.
- Logical operators are needed: && (and), || (or), ! (not).

A	B	A && B	A || B	!A
nonzero (true)	nonzero (true)	1(true)	1(true)	0 (false)
nonzero (true)	0 (false)	0 (false)	1(true)	0 (false)
0 (false)	nonzero (true)	0 (false)	1(true)	1(true)
0 (false)	0 (false)	0 (false)	0 (false)	1(true)

Operator Precedence

Operators

Function calls

! + − (unary operators)

*, /, %

+, -

==, >=, <=, >, <, !=

&&

||

=

Evaluation of Boolean Expressions

- The evaluation of a boolean expression proceeds according to the precedence and associativity of the operators.
- Example #1: What is the value of x?

```
int x, a = 4, b = -2, c = 0;
x = (a > b && b > c || a == b);
```

- Example #2: What is the value of x?

```
x = ((a > b) && !(b > c));
```

English Conditions as C Expressions

`x = 3.0 , y = 4.0 , and z = 2.0`

English Condition	Logical Expression	Evaluation
x and y are greater than z	`x > z && y > z`	1 && 1 is 1 (true)
x is equal to 1.0 or 3.0	`x == 1.0 || x == 3.0`	0 || 1 is 1 (true)
x is in the range z to y , inclusive	`z <= x && x <= y`	1 && 1 is 1 (true)

x is outside the range z to y	`!(z <= x && x <= y)`
	`z > x || x > y`
		!(1 && 1) is 0 (false)
		0 || 0 is 0 (false)

Short-circuit Evaluation

- Does the following code give an error if a is zero?

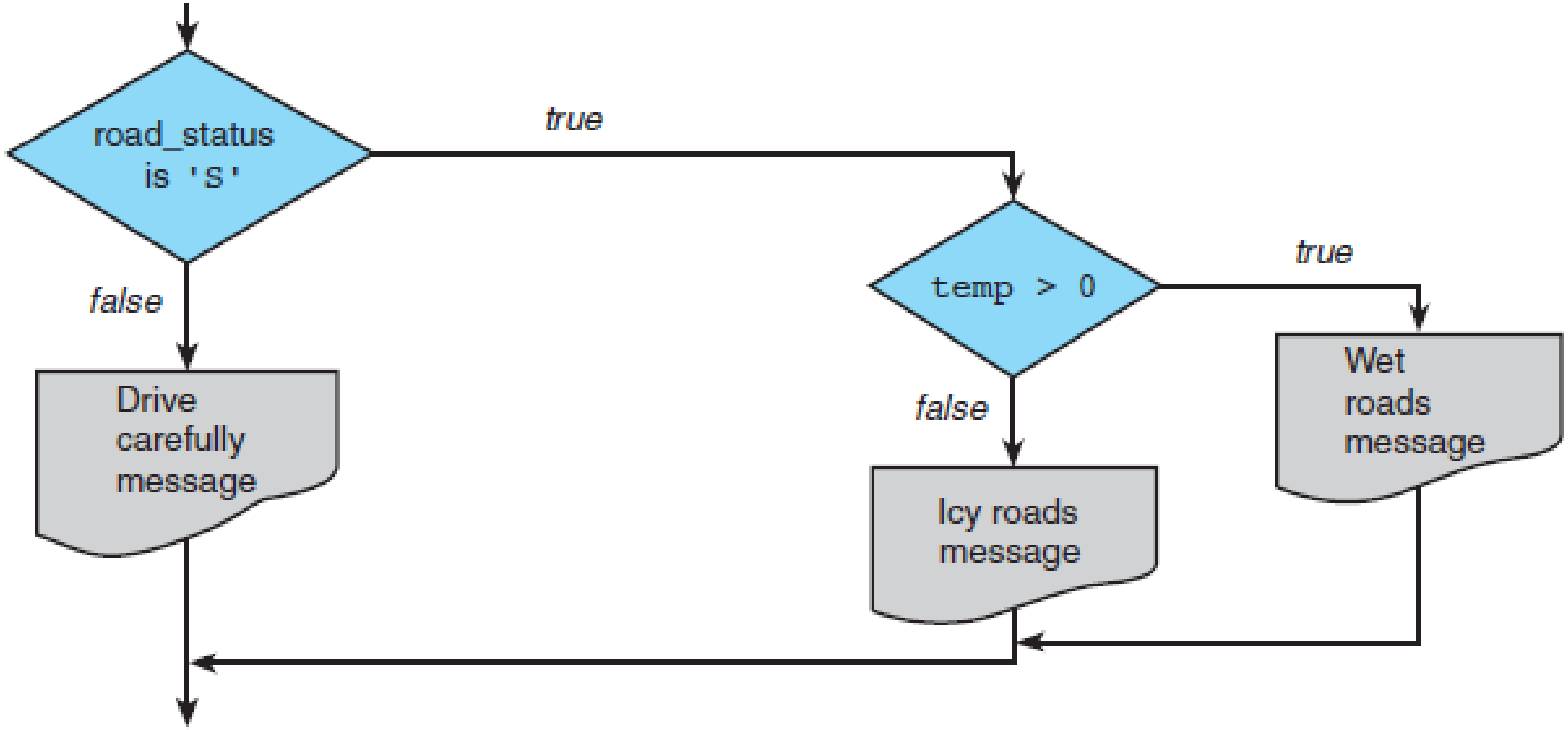
```
if ((a != 0) && (b/a > 3))
   printf(...);
```

- Short-circuit evaluation uses the following facts:
- expr1 || expr2 : If expr1 is true, skip evaluating expr2, as the result will always be true.

- expr1 && expr2: If expr1 is false, skip evaluating expr2, as the result will always be false.

Nested `if` Statements

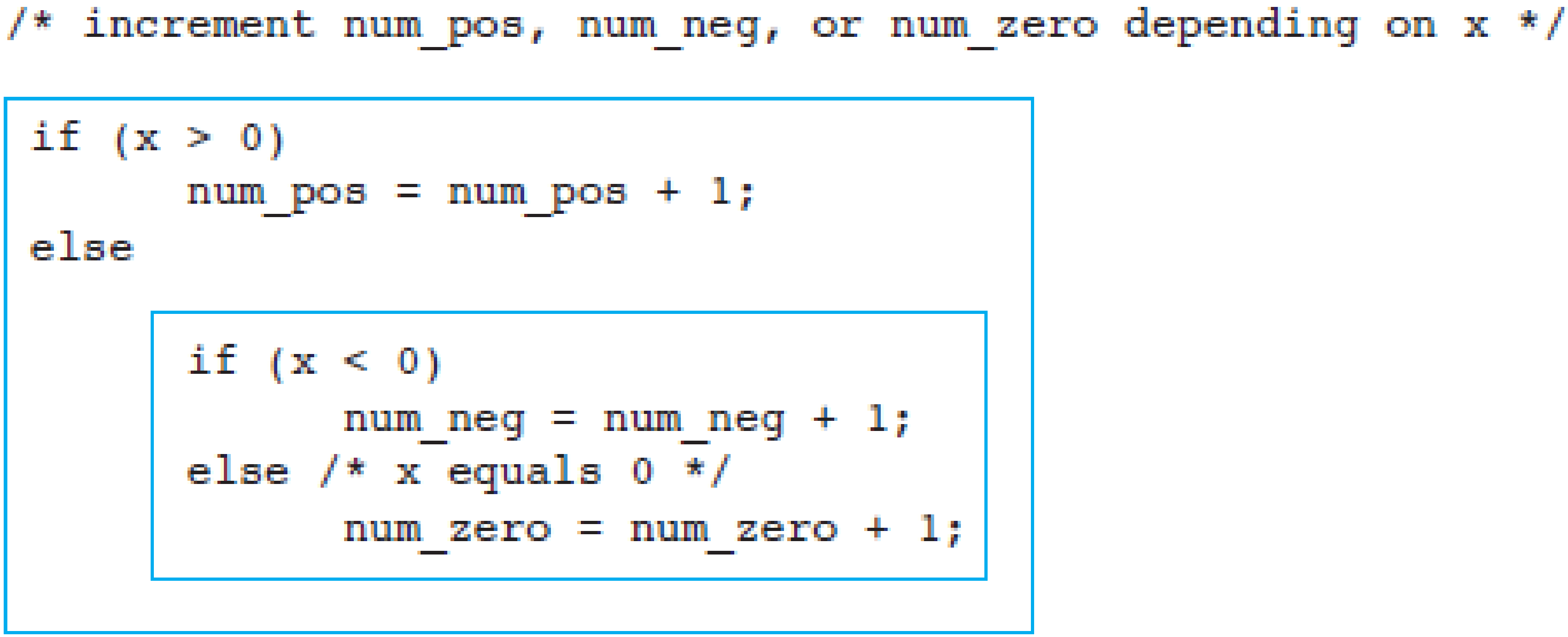
- An if statement with another if statement as its true task or its false task



Nested `if` Statements

```
if (road_status == 'S')
{
   if (temp > 0)
   {
      printf("Wet roads ahead\n");
      printf("Stopping time doubled\n");
   }
   else
   {
      printf("Icy roads ahead\n");
      printf("Stopping time quadrupled\n");
   }
}
else
   printf("Drive carefully!\n");
```

Nested **if** Statements



```
/* increment num_pos, num_neg, or num_zero depending on x */
if (x > 0)
      num_pos = num_pos + 1;
else
      if (x < 0)
            num_neg = num_neg + 1;
      else /* x equals 0 */
            num_zero = num_zero + 1;
```

Multiple-Alternative Decision Form of Nested `if`

- SYNTAX:

```
if ( condition 1 )
   statement 1
else if ( condition 2 )
   statement 2
   .
   .
   .
else if ( condition n )
   statement n
else
   statement e
```

Multiple-Alternative Decision Form of Nested **if**

- EXAMPLE: /* increment num_pos, num_neg, or num_zero depending on x */

```
if (x > 0)
   num_pos = num_pos + 1;
else if (x < 0)
   num_neg = num_neg + 1;
else /* x equals 0 */
   num_zero = num_zero + 1;
```

The **switch** Multiple-Selection Structure

```
switch ( integer expression )
{
    case constant1 :
        statement(s)
        break ;
    case constant2 :
        statement(s)
        break ;

    .  .  .
    default:
        statement(s)
        break ;
}
```

`switch` Statement Details

- The last statement of each case in the switch should almost always be a break.
- The break causes program control to jump to the closing brace of the switch structure.
- Without the break, the code flows into the next case. This is almost never what you want.
- A switch statement will compile without a default case, but always consider using one.

Good Programming Practices

- Include a default case to catch invalid data.
- Inform the user of the type of error that has occurred (e.g., “Error - invalid day.”).
- If appropriate, display the invalid value.
- If appropriate, terminate program execution

switch Example

```
switch ( day )
{
   case 0:  printf (“Sunday\n”) ;
        break ;
   case 1:  printf (“Monday\n”) ;
        break ;
   case 2:  printf (“Tuesday\n”) ;
        break ;
   case 3:  printf (“Wednesday\n”) ;
        break ;
   case 4:  printf (“Thursday\n”) ;
        break ;
   case 5:  printf (“Friday\n”) ;
        break ;
   case 6:  printf (“Saturday\n”) ;
        break ;
   default:  printf (“Error -- invalid day.\n”) ;
        break ;
}
```

Why Use a **`switch`** Statement?

- A nested if-else structure is just as efficient as a switch statement.
- However, a switch statement may be easier to read.
- Also, it is easier to add new cases to a switch statement than to a nested if-else structure.

Home-works

1. Write a C program that prompts the user to input tree integer values and find the greatest and smallest of the three values.

2. Write a program that determines a student's grade. The program will read three scores and determine the grade based on the following rules.

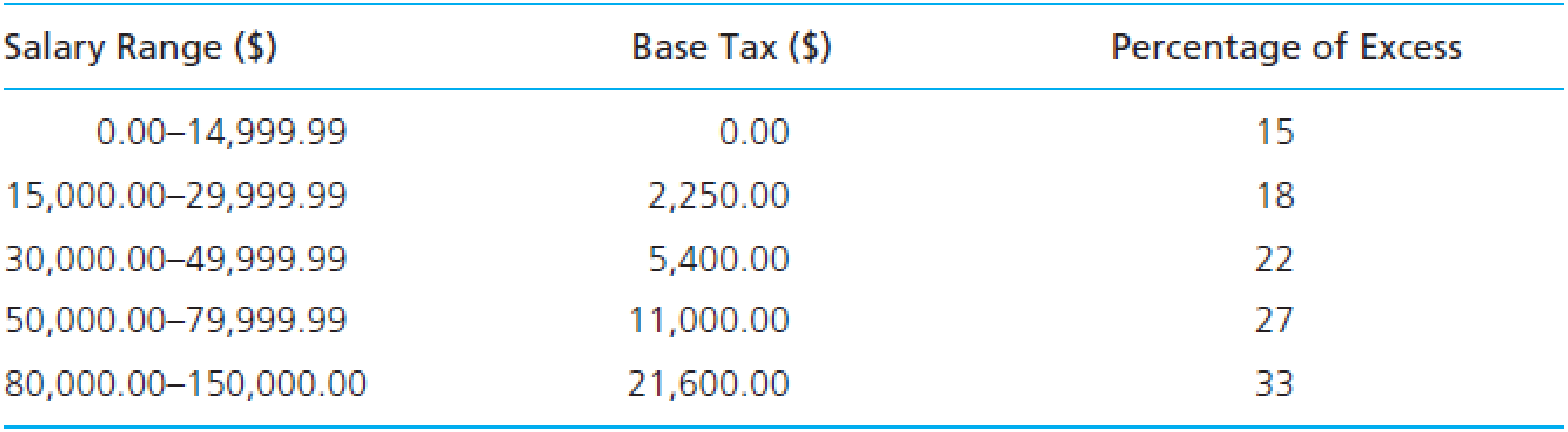
    if the average score is equal to or above 90%, grade = A

    if the average score is between 70% and 89.99%, grade = B

    if the average score is between 50% and 69.99%, grade = C

    if the average score is below 50%, grade = F

3. Calculate tax.



Salary Range ($)	Base Tax ($)	Percentage of Excess
0.00–14,999.99	0.00	15
15,000.00–29,999.99	2,250.00	18
30,000.00–49,999.99	5,400.00	22
50,000.00–79,999.99	11,000.00	27
80,000.00–150,000.00	21,600.00	33

Home-works

4. Determine if a year (provided as input) is a leap-year or not. Rule: A year is a leap year if it is perfectly divisible by four - except for the years which are divisible by 100 but not divisible by 400. for example, both 1996 and 2000 are leap years. But neither 1990 nor 1900 is a leap year.

5. Write a program to compute the real roots of a quadratic equation of the form . The program should prompt the user to enter the constants . The roots are calculated according to the rules:-

a) If is zero, there is only one root, which is .
b) If is negative, there are no real roots.
c) For all other cases, the two real roots are .