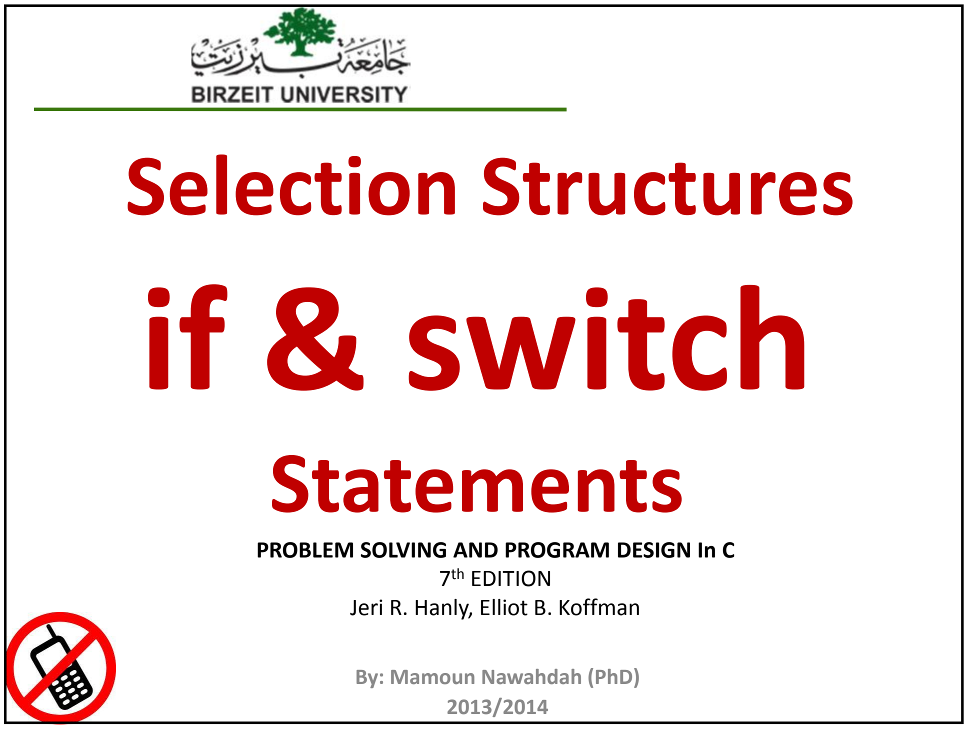


BIRZEIT UNIVERSITY

Selection Structures if & switch Statements

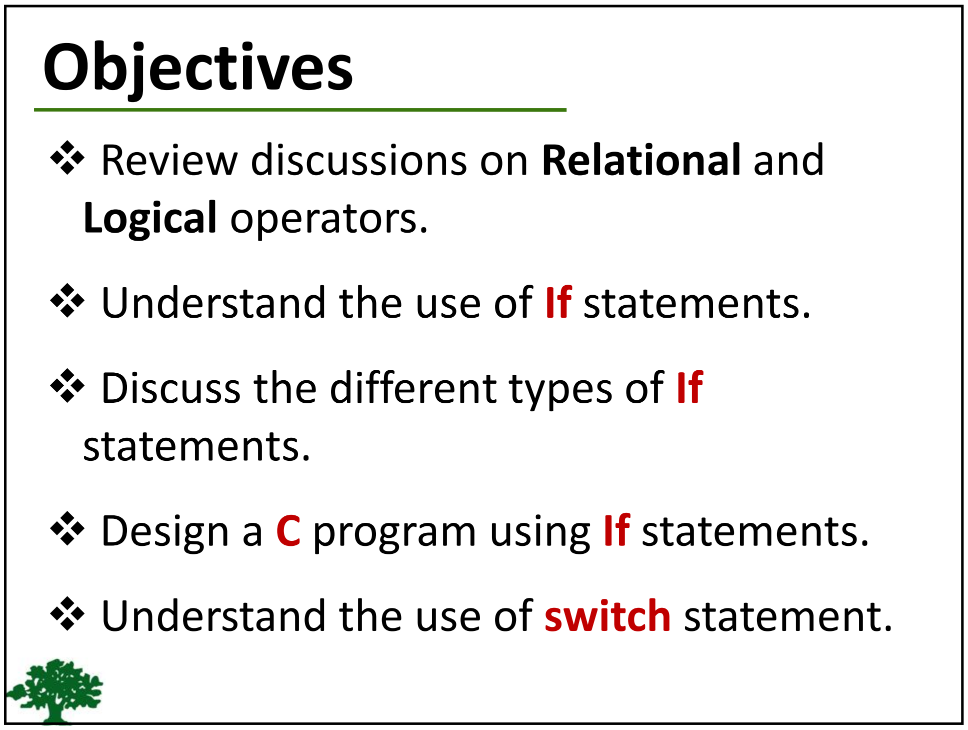
PROBLEM SOLVING AND PROGRAM DESIGN In C

7th EDITION

Jeri R. Hanly, Elliot B. Koffman

By: Mamoun Nawahdah (PhD)

2013/2014



Objectives

- ❖ Review discussions on **Relational** and **Logical** operators.
- ❖ Understand the use of **If** statements.
- ❖ Discuss the different types of **If** statements.
- ❖ Design a **C** program using **If** statements.
- ❖ Understand the use of **switch** statement.

Relational Operators (Boolean Expressions)

- ❖ Used to **compare** constants, variables, or expressions.
- ❖ The use of relational operators result to conditional expressions which gives the value of **true (1)** or **false (0)**.

Relational Operators

- ❖ The following are the relational operators used in **C** that can be used to compare **A** and **B** of any data type:

Operator	**Meaning**	**Example**
==	equal	A ==B
!=	not equal to	A != B
>	greater than	A > B
<	less than	A < B
>=	greater than or equal to	A >= B
<=	less than or equal to	A <= B

Examples

❖ Here are some examples given that **x=10** and **y=5** :

x < y	➔	result: 0 (false)
x > y	➔	result: 1 (true)
x >= y	➔	result: 1 (true)
x == y	➔	result: 0 (false)
x != y	➔	result: 1 (true)

Logical Operators

❖ Used to compare or combine conditional expressions/relational expressions even complex ones.

❖ There are three logical operators used by **C**:

and **&&** , or **||** , not **!**

❖ If an expression uses one or more of these operators, it is called ***logical expression***.

❖ The truth tables below show the result of logical operation when applied to its operands:

The && Operator (and)

❖ The truth tables below show the result of logical operation when applied to its operands:

operand 1	operand 2	operand1 && operand2
nonzero (true)	nonzero (true)	1 (true)
nonzero (true)	0 (false)	0 (false)
0 (false)	nonzero (true)	0 (false)
0 (false)	0 (false)	0 (false)

The || Operator (or)

operand 1	operand 2	operand1 || operand2
nonzero (true)	nonzero (true)	1 (true)
nonzero (true)	0 (false)	1 (true)
0 (false)	nonzero (true)	1 (true)
0 (false)	0 (false)	0 (false)

The ! Operator (not)

operand 1	!operand1
nonzero (true)	0 (false)
0 (false)	1 (true)

Logical Operators

- If we want a condition which is **true** if two expressions are both **true**, we use the logical **AND** operator ➔ **&&**
- For example:

```
if ( num >= 10  &&  num <= 20)
   printf("num is between 10 and 20 inclusive\n");
```

Logical Operators

❖ If we want a condition which is **true** if either or both of two expressions are **true**, we use the logical **OR** operator ➔ ||

❖ For example:

```
if ( num < 10 || num > 20 )
   printf("num is not between 10 and 20 inclusive\n");
```

Logical Operators

❖ If we want a condition which is **true** if an expression is **false**, we use the logical **NOT** operator ➔ !

❖ For example:

```
if ( !(num >= 10  &&  num <= 20) )
   printf("num is not between 10 and 20 inclusive\n");
```

Control Statement

- ❖ To change the execution order of program.
- ❖ As the method of controlling the execution order.
- ❖ Conditional Statement: **if**, **switch**
- ❖ Repeat Statement: **for**, **while**, **do-while**
- ❖ Branch Statement: **break**, **continue**, **return**

if Statement

- ❖ Performs an action or sequence of action **if** the condition is **true** or skips the action **if** the condition is **False**.
  - **If** it rains I will bring an umbrella.
  - **If** the hard disk is full I will erase some files.

if – else Statement

❖ Performs an action or sequence of action **if** the condition is **True** or perform *another* action if the condition is **False.**

- **If** it rains I will bring an umbrella, **else** I will bring a hat.
- **If** the hard disk is full I will erase some files, **else** I will save more files.

if-else Statement -Syntax

```
if (condition) {
  statement1;
} else {
  statement2;
}
```

```
if (condition1) {
  statement1;
} else if (condition2) {
  statement2;
} else if (condition3) {
  statement3;
} else {
  statement4;
}
```

Example

```
#include <stdio.h>
int main ( ){
    int a , b ;
    printf ("Enter values for a and b : ") ;
    scanf ("%d%d",  &a,  &b ) ;
    if ( a < b ) {
           printf ("a is less than b\n") ;
    }  else if ( a == b ) {
                      printf (" a is equal to b\n") ;
           }  else {
                      printf ("a is larger than b\n") ;
           }
}
```

Example

```
#include <stdio.h>
int main ( ) {
     int score ;
     printf ("enter your test score:") ;
     scanf ("%d", &score)  ;
     if (score >= 90){
            printf ("Your score of %d is a A\n", score) ; }
     else if (score >= 80 && score < 90) {
                printf ("Your score of %d is a B\n", score) ; }
            else if (score >= 70) {
                      printf ("Your score of %d is a C\n", score) ; }
                 else if (score >= 60) {
                          printf ("Your score of %d is a D\n", score) ; }
                      else {
                                printf ("Your score of %d is an E\n", score) ; }
}
```

Programming Problems

❖ Write a **C** program that checks **if** an integer is **even** or **odd** number:

- **If** the integer is even display the message

  "***The number is even***".

- **If** the integer is odd display the message

  "***The number is odd***".

- **If** the number is zero (0) display the message

  "***Enter a non-zero number***".

Programming Problems

❖ Write a **C** program that accepts **5** grades of a student. Compute the **average** and display the letter equivalent of his average based on the following criteria:

Average	**Letter**
• 95 – 100	A+
• 90 – 94	A –
• 85 – 89	B +
• 80 – 84	B –
• 75 – 79	C
• below 75	F

Switch Case

- A multiple selection structure is useful when an algorithm contains a series of decisions in which a variable or expression is tested separately for one of several possible **integral** values.
- Each **integral** value represents a different action to be taken in the algorithm.
- C provides the switch multiple selection structure to implement this type of decision making.

switch-case Structures

- The switch - case syntax is:

```
switch (expression test) {
   case  case1_fixed_value :
               action(s) ;
               break;
   case  case2_fixed_value :
               action(s) ;
               break;
   default :
               action(s) ;
}
```

Note use of colon!

switch-case Structures

- The **switch** is the **“**controlling expression**”**.
- Can only be used with ***constant integer*** expressions.
  - Remember, a single character is a small positive integer.
- The expression appears in **( )**
- The case is a “**label”**.
- The label **must** be followed by a " **:** "
- Braces, **{ }** , not required around statements.
- **break;** used in either a repetition structure or a selection structure to **break out** of (to exit from) the structure.

A Sample Program to Illustrate switch-case

```
#include <stdio.h>
int main ( ) {
    char grade ;
    printf ("Enter your current letter grade\n") ;
    scanf(“%c”, &grade);
    switch (grade) {
        case ('a') :
        case ('A') :
                printf ("Good Job!\n") ;
                break;
        case ('b') :
        case ('B') :
                printf ("Pretty good.\n") ;
                break;
```

Continue →

```
                    case ('c') :
                    case ('C') :
                              printf ("Better get to work.\n") ;
                              break;
                    case ('d') :
                    case ('D') :
                              printf ("You are in trouble.\n") ;
                              break;
                    default :
                              printf ("You are failing!!\n") ;
              }  /* End of switch-case structure */

          }  /* End of main program  */
```

Use switch

❖ Write a C program that accepts two integer numbers. If the user press:

1 → Add the two numbers.

2 → Subtract the 1st integer from the 2nd integer.

3 → Multiply the two numbers.

4 → Divide the 1st integer from the 2nd integer.

if vs. switch - using if/else if/else

```
if (color == 1) {
    printf("The color is blue");
}
else if (color ==  2 ) {
                     printf("The color is red");
          }
          else if (color ==  3) {
                               printf("The color is green");
                     }
                     else {
                                printf(" unknown color");
                     }
```

if vs. switch - using swtich

```
switch (color)  {
    case  1:
           printf("The color is blue");
           break;
    case  2:
           printf("The color is red");
           break;
    case  3:
           printf("The color is green");
           break;
    default  :
           printf(“unknown color");
}
```

if vs. switch

- A **switch** statement does much the same job as an **if** statement, but it is more appropriate for situations where you have **many choices**, rather than only a few.

Exercises

1. Write an **If** statement to determine **if** an integer number entered is **positive** or **negative** number.
   - Convert the program to **switch** statement.
2. Read an integer value. Assume it is the number of a month of the year; print out the name of that month. Use **if** statement or **switch** statement.
3. Write a **switch** statement to determine if the letter stored in a variable is a vowel or consonant. Increment the **vowelCount** if it is a vowel otherwise increment the **consonantCount**.