

C Conditional Statements

Using Control-Flow Logic in C

SoftUni Team
Technical Trainers
Software University
<http://softuni.bg>



Table of Contents

1. Comparison and Logical Operators
2. The **if** Statement
3. The **if-else** Statement
4. Nested **if** Statements
5. The **switch-case** Statement



Comparison and Logical Operators

Category	Operators
Arithmetic	+ - * / % ++ --
Logical	&& ^ !
Comparison	== != < > <= >=
Assignment	= += -= *= /= %= &= = ^= <<= >>=

Comparison Operators

Operator	Notation in C#	Applicable for
Equals	==	integer types, symbols
Not Equals	!=	
Greater Than	>	integer types, symbols, pointers
Greater Than or Equals	>=	
Less Than	<	
Less Than or Equals	<=	

- Example:

```
int result = (5 <= 6);  
printf("%d", result); // 1
```

Logical Operators

Operator	Notation in C#	Example
Logical NOT	!	!false → true
Logical AND	&&	true && true → true
Logical OR		true false → true
Logical Exclusive OR (XOR)	^	true ^ false → true

■ De Morgan laws

- **!!A ⇔ A**
- **!(A || B) ⇔ !A && !B**
- **!(A && B) ⇔ !A || !B**



if and if-else

Implementing Conditional Logic



The `if` Statement

- The simplest conditional statement
- Enables you to test for a condition
- Branch to different blocks in the code depending on the result
- The simplest form of the `if` statement:

```
if (condition)
{
    statements;
}
```

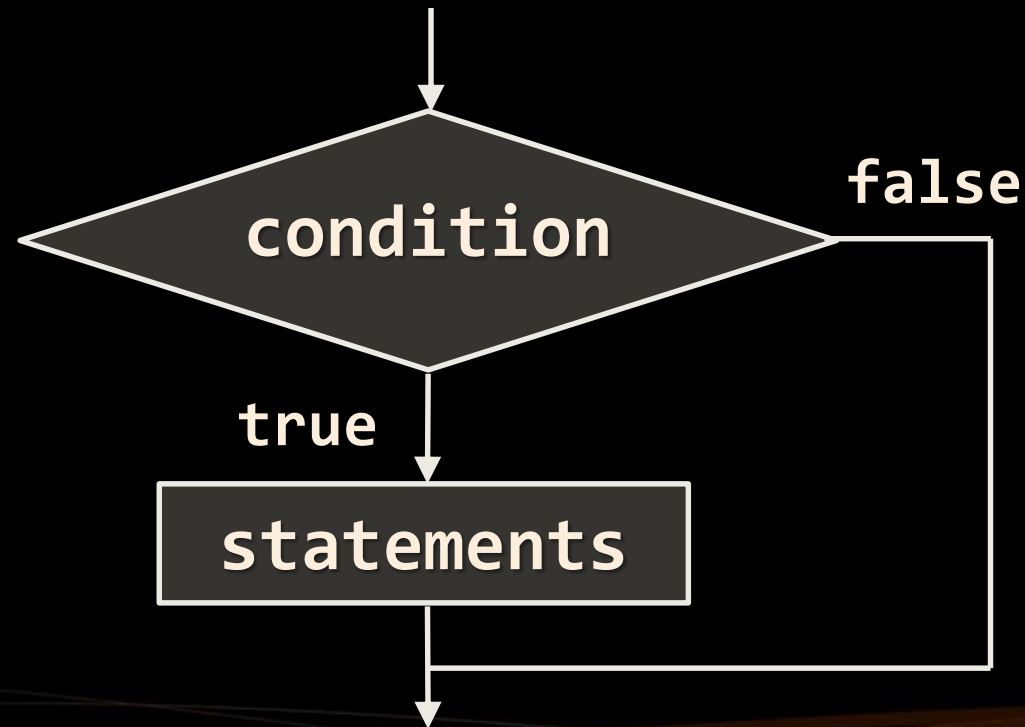
Condition and Statement

- The **condition** can be:
 - Integer variable – **0** for false, **1** for true
 - Boolean logical expression (evaluates to 0 or 1)
 - Comparison expression (evaluates to 0 or 1)
 - Every other type (**0.0**, '**\0**', **NULL** evaluate to 0)
- The **statement** can be:
 - Single statement ending with a semicolon
 - Block enclosed in curly braces



How It Works?

- The **condition** is evaluated
 - If it is true, the statement is executed
 - If it is false, the statement is skipped



The if Statement – Example

```
int main()
{
    int biggerNum, smallerNum;
    printf("Enter two numbers:\n");
    scanf("%d", &biggerNum);
    scanf("%d", &smallerNum);
```

Pass the address of
biggerNum

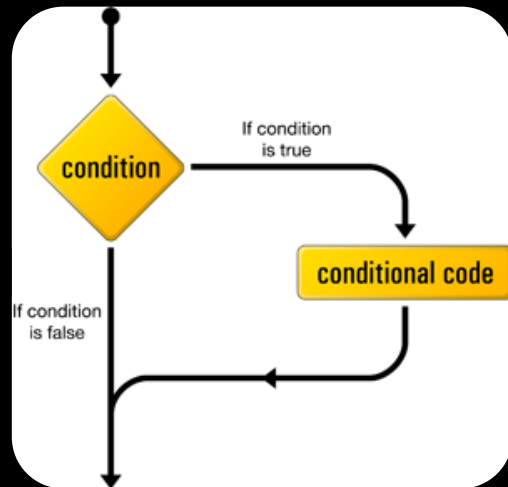
```
    if (smallerNum > biggerNum)
    {
        biggerNum = smallerNum;
    }
```

Evaluates to **0** or **1**

```
    printf("The greater number is %d.", biggerNum);
    return 0;
}
```

The `if` Statement

Live Demo



`if` ?

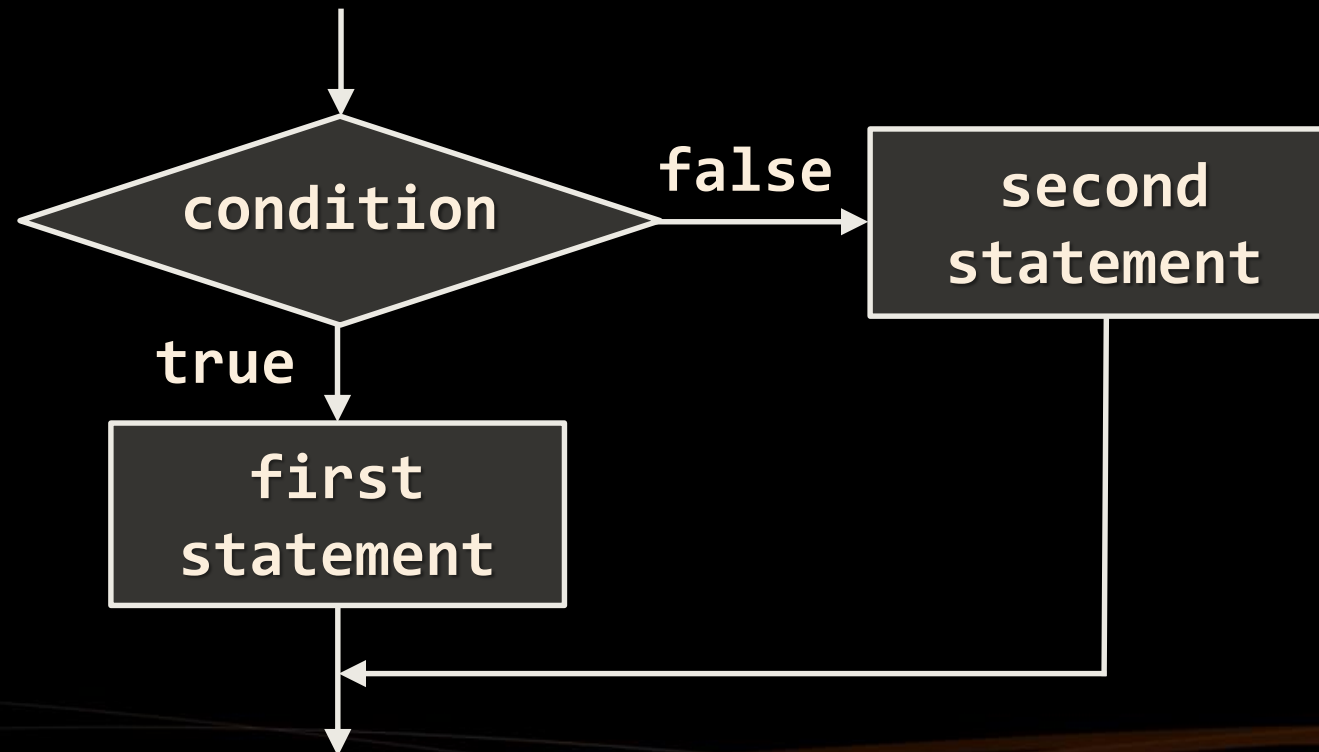
The if-else Statement

- More complex and useful conditional statement
- Executes one branch if the condition is true, and another if it is false
- The simplest form of an **if-else** statement:

```
if (expression)
{
    some_statement;
}
else
{
    another_statement;
}
```

How It Works?

- The **condition** is evaluated
 - If it is true, the first statement is executed
 - If it is false, the second statement is executed



if-else Statement – Example

- Checking a number if it is odd or even

```
int num;  
scanf("%d", num);  
  
if (num % 2 == 0)  
{  
    printf("This number is even.");  
}  
else  
{  
    printf("This number is odd.");  
}
```

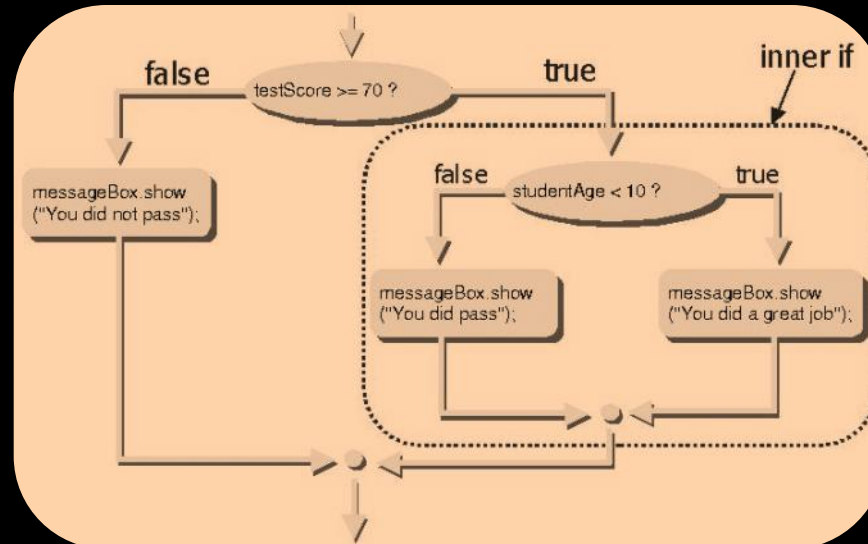

The if-else Statement

Live Demo



Nested if Statements

Creating More Complex Logic



Nested if Statements

- **if** and **if-else** statements can be **nested**
 - Used inside one inside another
- Each **else** corresponds to its closest preceding **if**

```
if (expression)
{
    if (expression)
        some_statement;
    else
        another_statement;
}
else
    third_statement;
```


Nested if – Good Practices

- Always use `{ ... }` blocks to avoid ambiguity
 - Even when a single statement follows
- Avoid using more than three levels of nested **if** statements
- Put the case you normally expect to process first
 - Then write the unusual cases
- Arrange the code to make it more readable

Nested if Statements – Example

```
if (first == second)
{
    printf("These two numbers are equal.");
}
else
{
    if (first > second)
    {
        printf("These first number is bigger.");
    }
    else
    {
        printf("These second number is bigger.");
    }
}
```

Nested if Statements

Live Demo



```
if (expression)
{
    if (expression)
    {
        statement;
    }
    else
    {
        statement;
    }
}
```

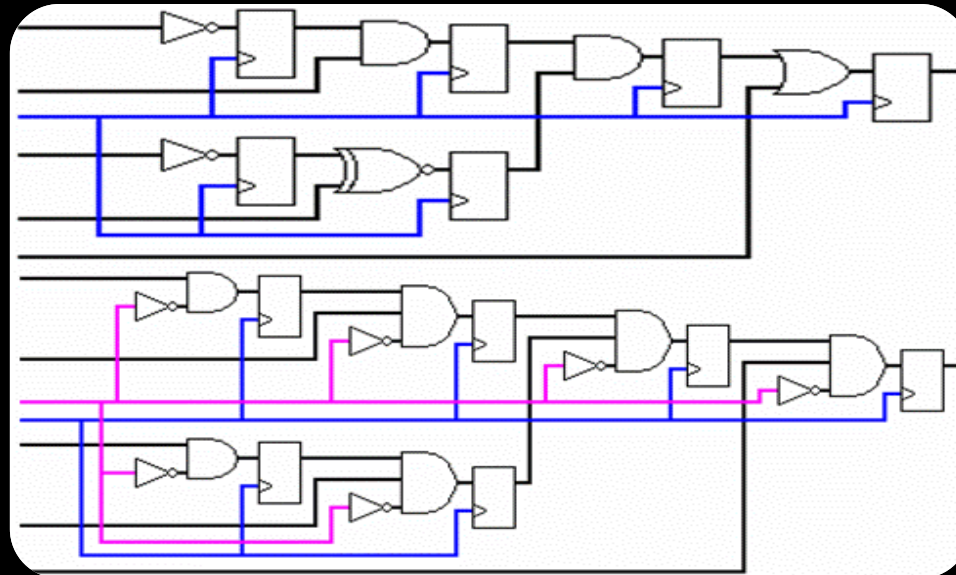

Multiple if-else-if-else...

- We may need to use another **if**-construction in the **else** block
 - Thus **else if** can be used:

```
int ch = 'X';
if (ch == 'A' || ch == 'a')
{
    printf("Vowel [ei]");
}
else if (ch == 'E' || ch == 'e')
{
    printf("Vowel [i:]");
}
else if ...
else ...
```

Multiple if-else Statements

Live Demo



switch-case

Performing Several Comparisons at Once



The switch-case Statement

- Selects for execution a statement from a list depending on the value of the **switch** expression

```
switch (day)
{
    case 1: printf("Monday"); break;
    case 2: printf("Tuesday"); break;
    case 3: printf("Wednesday"); break;
    case 4: printf("Thursday"); break;
    case 5: printf("Friday"); break;
    case 6: printf("Saturday"); break;
    case 7: printf("Sunday"); break;
    default: printf("Error!"); break;
}
```


How switch-case Works?

1. The **expression** is evaluated
2. When one of the constants specified in a case label is equal to the expression
 - The **statement** that corresponds to that case is executed
3. If no case is equal to the expression
 - If there is **default case**, it is executed
 - Otherwise the control is transferred to the end point of the switch statement



The switch-case Statement

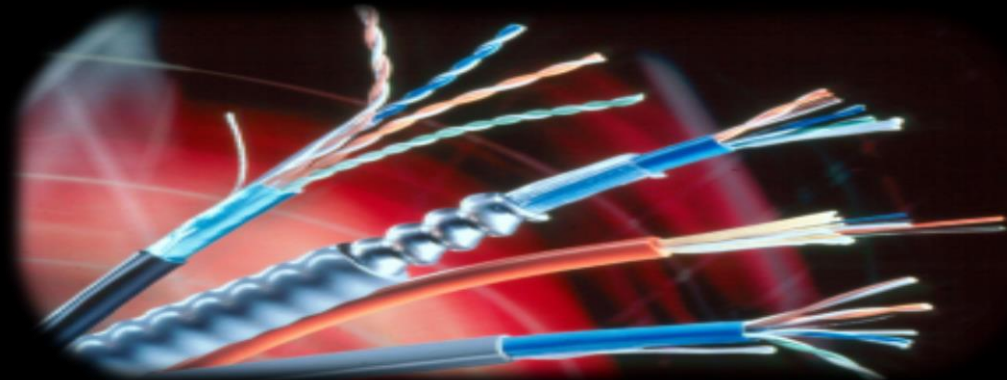
Live Demo

Using switch: Rules

- Variable types like integral types and enums can be used for **switch** expression
- The keyword **break** exits the switch statement
- Be careful with "falling-through" multiple cases!
 - Use **break** after each case
- Multiple labels that correspond to the same statement are permitted

Multiple Labels in a switch-case

Live Demo



Using switch – Good Practices

- There must be a separate **case** for every normal situation
- Put the normal case first
 - Put the most frequently executed cases first and the least frequently executed last
- Order cases alphabetically or numerically
- In **default** use case that cannot be reached under normal circumstances

Summary

- Comparison and logical operators are used to compose logical conditions
- The conditional statements **if** and **if-else** conditionally execution of blocks of code
 - Constantly used in computer programming
 - Conditional statements can be **nested**
- The **switch** statement easily and elegantly checks an expression for a sequence of values



C Programming – Conditional Statements



Questions?



License

- This course (slides, examples, demos, videos, homework, etc.) is licensed under the "Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International" license



- Attribution: this work may contain portions from
 - "Programming Basics" course by Software University under CC-BY-SA license

Free Trainings @ Software University

- Software University Foundation – softuni.org
- Software University – High-Quality Education, Profession and Job for Software Developers
 - softuni.bg
- Software University @ Facebook
 - facebook.com/SoftwareUniversity
- Software University @ YouTube
 - youtube.com/SoftwareUniversity
- Software University Forums – forum.softuni.bg

