

Lecture 5

Selection (Making Decision)

Today's Topics

- Logical data and operators
- **if** statement
- **switch** statement

Logical data

- Also called **Boolean**
- In C, there is no True or False Value.
- A **Zero value** is treated as **False**
- A **Non-Zero** value is treated as **True**

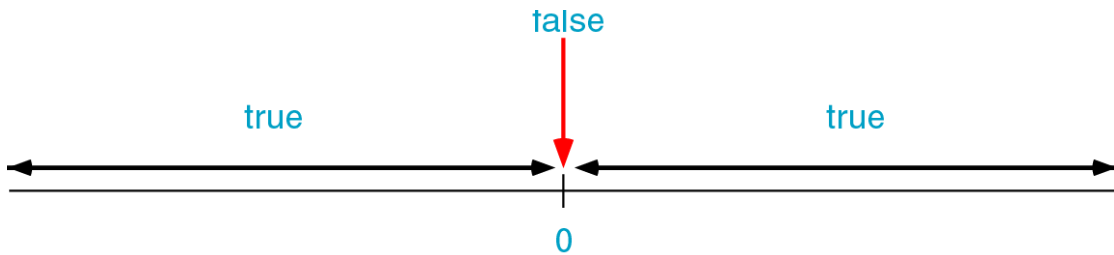


Figure 5-1: True and False in arithmetic scale

Figure 5-2: Logical operators truth table

not

x	!x
false	true
true	false

logical

!

x	!x
zero	1
nonzero	0

C Language

and

x	y	x&& y
false	false	false
false	true	false
true	false	false
true	true	true

logical

&&

x	y	x&& y
zero	zero	0
zero	nonzero	0
nonzero	zero	0
nonzero	nonzero	1

C Language

or

x	y	x y
false	false	false
false	true	true
true	false	true
true	true	true

logical

||

x	y	x y
zero	zero	0
zero	nonzero	1
nonzero	zero	1
nonzero	nonzero	1

C Language

Figure 5-3: Operations for logical and/or

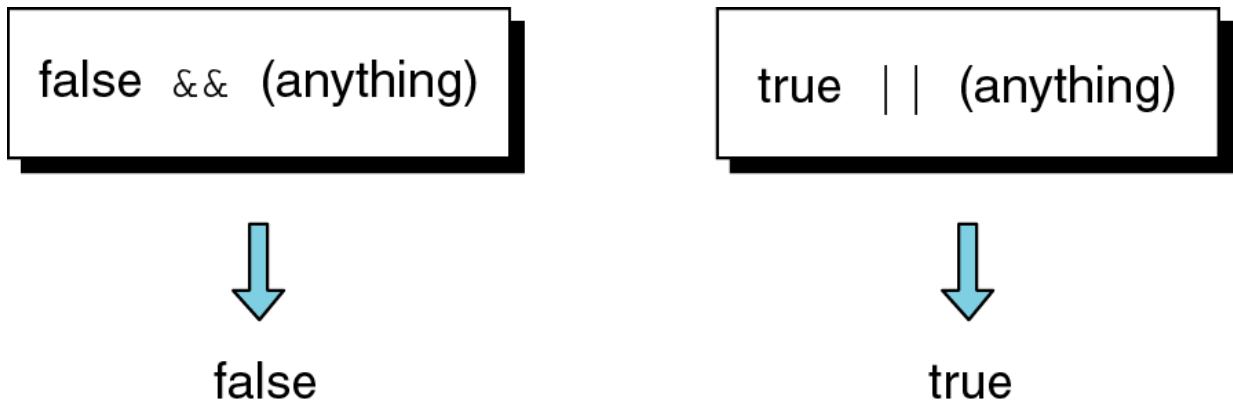
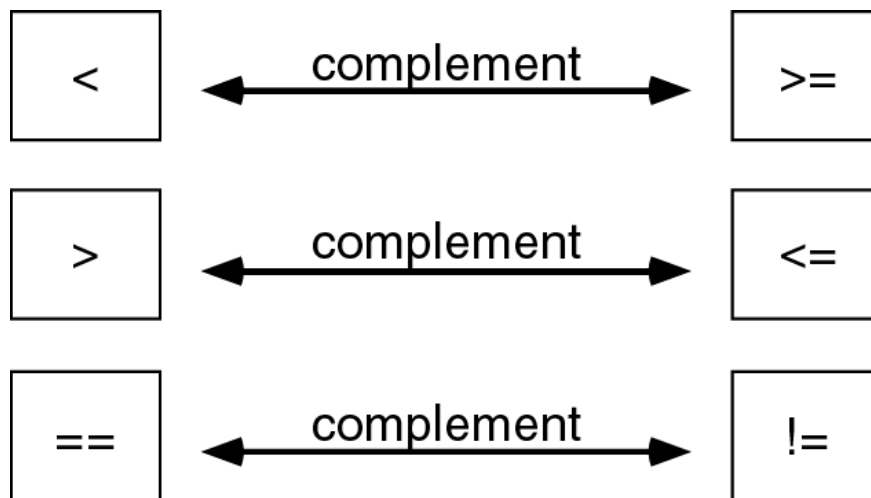


Figure 5-4: Relational operators

Operator	Meaning	Precedence
<	less than	10
<=	less than or equal	
>	greater than	
>=	greater than or equal	
==	equal	9
!=	not equal	



Making decisions

- Sometimes your programs need to make logical choices.
- Example:

IF score is higher than 50

THEN grade is PASS


ELSE grade is FAIL

- In C, this corresponds to `if` statement with three parts:

```
if (score > 50) //part 1
{
    grade = PASS; //part 2
}
else
{
    grade = FAIL; //part 3
}
```

if statement


- Part 1 : the **condition** - an expression that is evaluated to **TRUE** or **FALSE**.



```
if (score > 50)
{
    grade = PASS;
}
else
{
    grade = FAIL;
}
```

if statement

- Part 2 : the **TRUE-PART** - a block of statements that are executed if the condition evaluates to **TRUE**

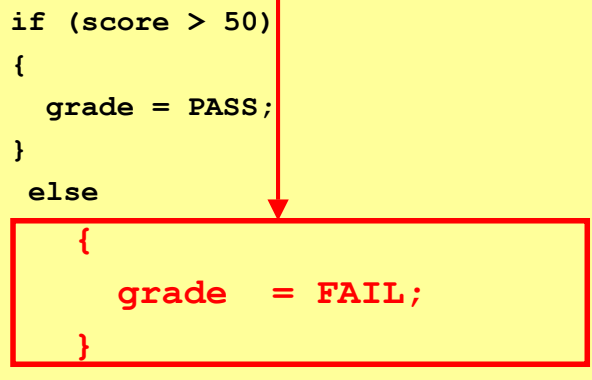


```
if (score > 50)
{
    grade = PASS;
}
else
{
    grade = FAIL;
}
```

if statement

- Part 3 : the **FALSE-PART** - a block of statements that are executed if the condition evaluates to **FALSE**

```
if (score > 50)
{
    grade = PASS;
}
else
{
    grade = FAIL;
}
```



if the condition evaluates to FALSE, the TRUE-PART is skipped.

if statement

- Sometimes there is no FALSE-PART:

```
if ( attendance < 0.8 )
{
    exam_grade = FAIL;
}
```

if statement

- If the TRUE-PART (or FALSE-PART) consists of only **one statement**, then the curly braces may be omitted.
- Example: these two statements are equivalent:

```
if (score > 50)
{
    grade = PASS;
}
else
{
    grade = FAIL;
}
```

```
if (score > 50)
    grade = PASS;
else
    grade = FAIL;
```

if statement

- Sometimes there are more than two parts. In those cases you may use **cascading (or nested) if/else** statements:

```
if (score > 90)
    letter_grade = 'A';
else if (score > 75)
    letter_grade = 'B';
else if (score > 60)
    letter_grade = 'C';
else if (score > 50)
    letter_grade = 'D';
else
    letter_grade = 'F';
```

if statement

- Three forms of **if** statements are shown at the next table.
- The *condition* is always in parentheses
- All TRUE-PARTS and all FALSE-PARTS are a single statement or a **block** of statements (also called **compound statement**)

```
if(condition)
    statement;
```

```
if (condition)
{ statement;
  |
  |
  |
  statement;
}
```

```
if (condition)
{ statement;
  |
  |
  |
  statement;
}
else
{ statement;
  |
  |
  |
  statement;
}
```

- A **compound statement** is a one or more statements that are grouped into a single block. It is enclosed with the bracket symbols, **{}**.

- Example:

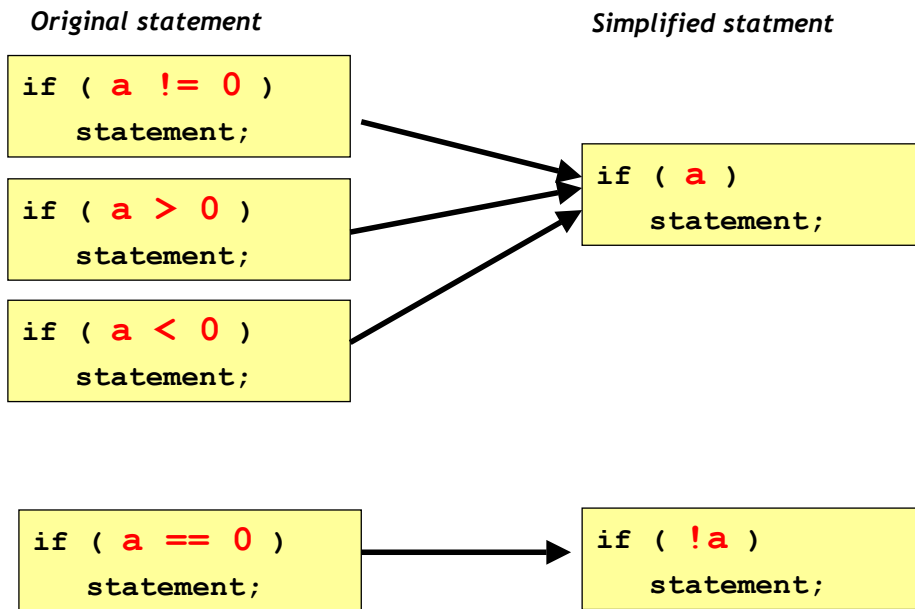
```
if (value>0)
    printf("%d",value); ← a single statement

value = value * 2; ← a single statement

if (value> 10)
{
    value = 10;
    printf("%d",value);
} ← a compound statement
```

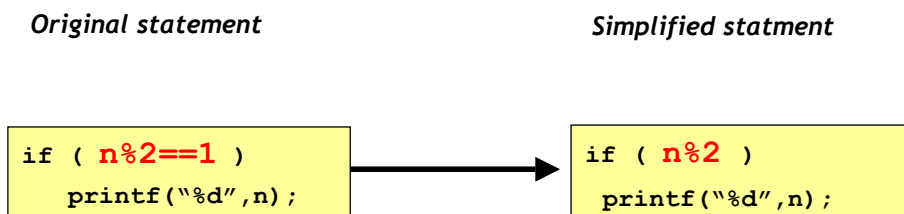
Simplifying if statements

- Simplifying the condition:

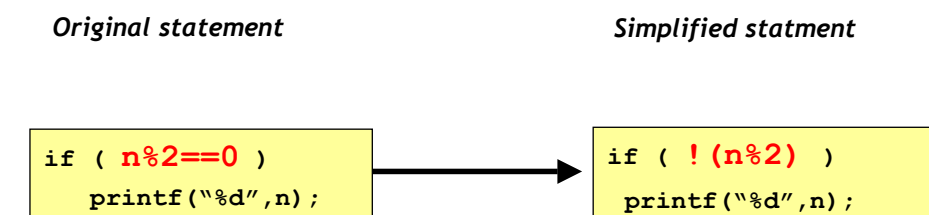


Simplifying if statements

- Example 1 : print a number only if it is an **odd** number

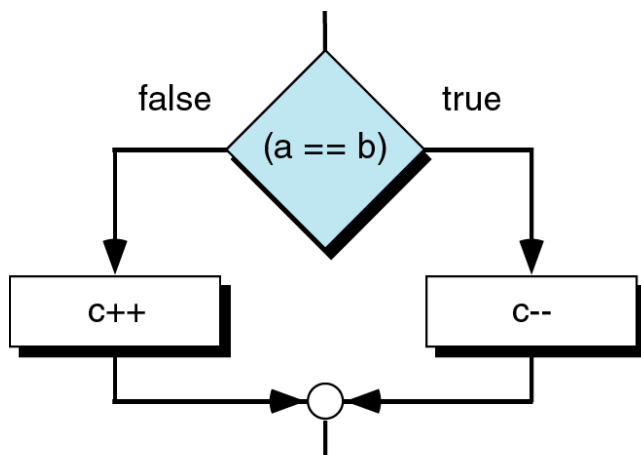


- Example 2: print a number only if it is an **even** number

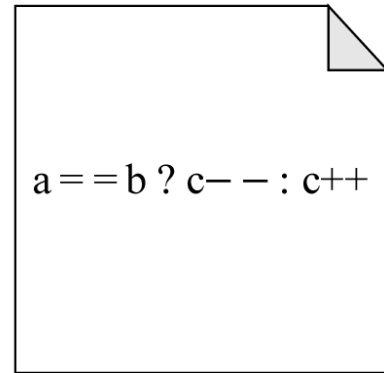


Simplifying if statements

- Conditional Expressions:



(a) Logic Flow



(b) Code

Simplifying if statements

- Conditional Expressions:

Syntax:

```
condition ? value1 : value2
```

If the condition is
True, take the value1

If the condition is
False, take the value2

Example:

```
p = (q < 5) ? q + 1 : 5;
```

This statement
means

```
if (p < 5)
    p = q + 1;
else
    p = 5;
```

switch statement

- If there are many nested if/else statements, you may be able to replace them with a switch statement:

```
if (letter_grade == 'A')
    printf("Excellent!");
else if (letter_grade == 'B')
    printf("Very good!");
else if (letter_grade == 'C')
    printf("Good");
else if (letter_grade == 'D')
    printf("Adequate");
else
    printf("Fail");
```



```
switch (letter_grade)
{
    case 'A' : printf("Excellent!");
              break;

    case 'B' : printf("Very good!");
              break;

    case 'C' : printf("Good");
              break;

    case 'D' : printf("Adequate");
              break;

    default  : printf("Fail");
              break;
}
```

switch statement

```
switch (expression)
{
    case value1: statements_1;
                break;

    case value2 : statements_2;
                break;

    ...

    default : statements;
            break;
}
```

How the **switch** statement works?

1. Check the value of **expression**.
2. Is it equal to **value1**?
 - If yes, execute the **statements_1** and **break** out of the switch.
 - If no, is it equal to **value2**? etc.
3. If it is not equal to any values of the above, execute the **default statements** and then **break** out of the switch.

switch statement

Example:

```
int value = 1;
switch (value)
{
  case 1: printf("One");
          break;
  case 2: printf("Two");
          break;
  default : printf("Neither One nor Two");
            break;
}
```

evaluates to 1

Prints One

break out of the switch

it is equal to this case-value (i.e. $1=1$). So, execute the statements of the case 1.

Output:

One

switch statement

Example:

```
int value = 1;
switch (value + 1)
{
  case 1: printf("One");
          break;
  case 2: printf("Two");
          break;
  default : printf("Neither One nor Two");
            break;
}
```

this expression evaluates to 2

Prints Two

break out of the switch

it is not equal to this case-value (i.e. $2 \neq 1$). So, skip the statements of case 1 and move to the next case.

it is equal to this case-value (i.e. $2=2$). So, execute the statements of the 'case 2'.

Output:

Two

switch statement

Example:

```
int value = 5;
switch (value)
{
    case 1: printf("One");
            break;
    case 2: printf("Two");
            break;
    default : printf("Neither One nor Two");
              break;
}
```

The switch expression (i.e. 5) is not equal to both cases (i.e. $5!=1$ and $5!=2$). So, their statements are skipped.

When the 'default case' is reached, its statements are always executed.

evaluates to 5

Prints Neither One nor Two

break out of the switch

Output:

Neither One nor Two

switch statement

What if the break statement is not written?

```
int value = 1;
switch (value)
{
    case 1: printf("One\n");
    case 2: printf("Two\n");
            break;
    default : printf("Neither One nor Two\n");
              break;
}
```

It is equal to this case-value (i.e. $1==1$). So, execute the statements of the case 1.

evaluates to 1

Prints One

No break statement here. So, no break out and move to the next line.

Prints Two

break out of the switch

Output:

One
Two

switch statement

- The switch expression must be an `int` or a `char`.
- The following examples would be an error

```
void main()
{
    float point=4.0;
    int mark;

    switch (point)
    {
        case 4 : mark = 100;
                break;

        case 3.7 : mark = 80;
                break;

        default : mark = 0;
                break;
    }
}
```

Error! The switch expression cannot be a float value

```
void main()
{
    char name[]="Ali";
    int mark;

    switch (name)
    {
        case "Ali" : mark=95;
                    break;

        case "Aminah": mark=90;
                    break;

        default : mark=50;
                break;
    }
}
```

Error! The switch expression cannot be a string value

switch statement

- The case-value must be a constant (literal, memory or defined constant)
- The following example would be an error

```
void main()
{
    #define DEFINE 1
    const int const2=2;
    int var3 = 3;
    int value;

    switch (value)
    { case 0 : printf("Four");
      case DEFINE : printf("One");
      case const2 : printf("Two");
      case var3 : printf("Three");
    }
}
```

a literal is OK

a defined constant is OK

a memory constant is OK

Error! case-value cannot be a variable