Conditional Statements

Prepared by: Lec Tasmiah Tamzid Anannya, CS Dept, AIUB

Conditional Statements

- A *conditional statement* lets us choose which statement will be executed next
- Thereforethey are sometimes called *selection statements*
- Conditional statements give us the power to make basic decisions
- The C conditional statements are the:
  - *if statement*
  - *if-else statement*
  - *switch statement*

The if Statement

- The *if statement* has the following syntax:

The if Statement

- The *if statement* has the following syntax:

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

The if Statement

- The *if statement* has the following syntax:

`if` is a C reserved word

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

The if Statement

- The *if statement* has the following syntax:

The *condition* must be a boolean expression. It must evaluate to either true or false.

if is a C reserved word

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

The if Statement

- The *if statement* has the following syntax:

The *condition* must be a boolean expression. It must evaluate to either true or false.

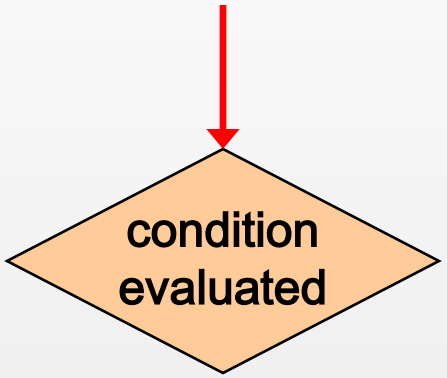
if is a C reserved word

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

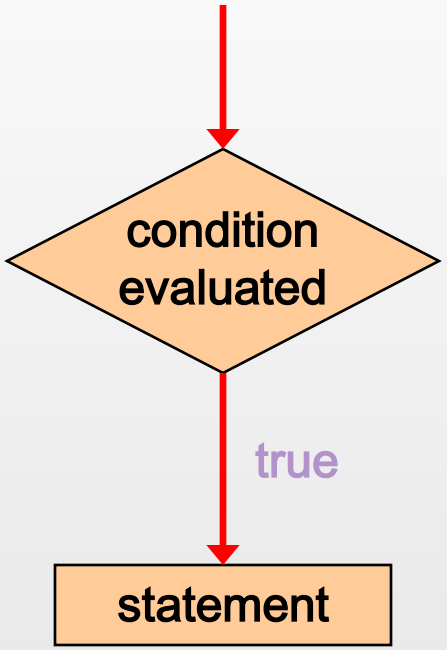
If the *condition* is true, the *statement* is executed. If it is false, the *statement* is skipped.

Logic of an if statement

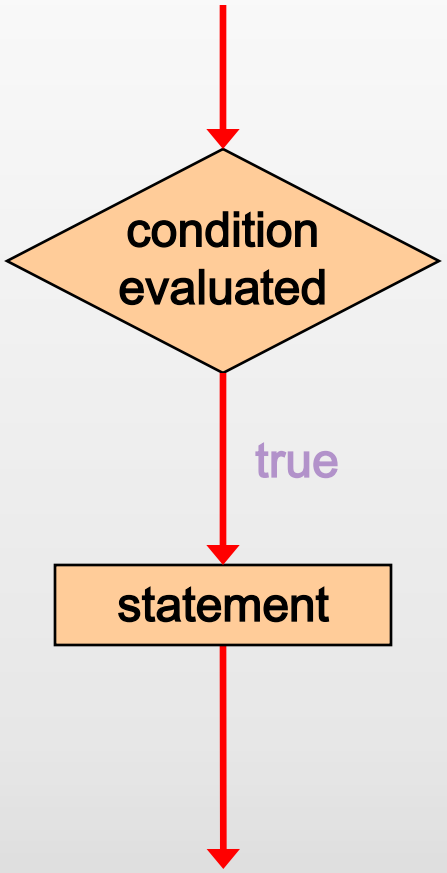
Logic of an if statement



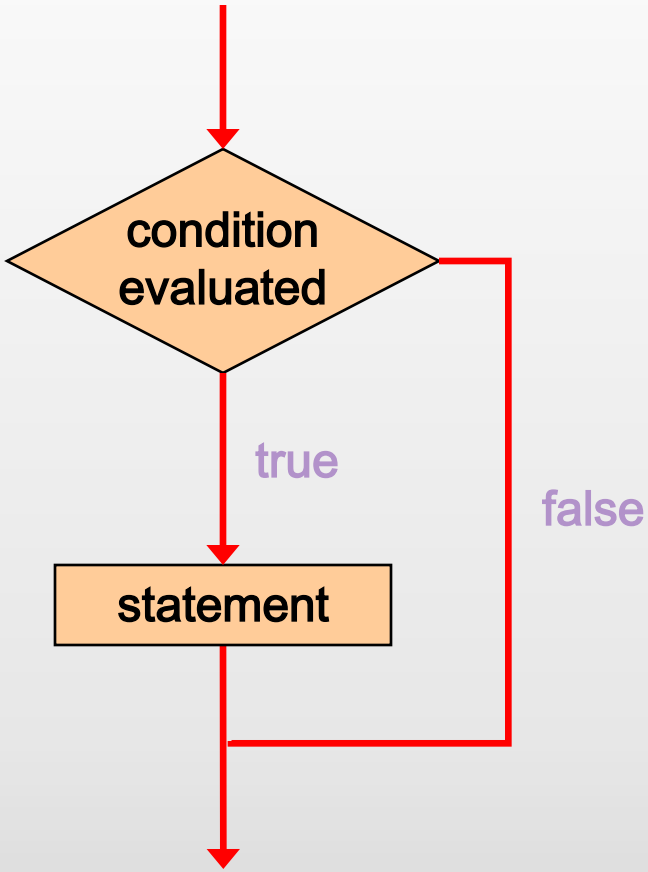
Logic of an if statement



Logic of an if statement



Logic of an if statement



Relational Operators

- A condition often uses one of C's *equality operators* or *relational operators*

==	equal to
!=	not equal to
<	less than
>	greater than
<=	less than or equal to
>=	greater than or equal to

- Note the difference between the equality operator (==) and the assignment operator (=)

The if Statement

- An example of an `if` statement:

The if Statement

- An example of an `if` statement:

```
if (sum > MAX)
{
   delta = sum - MAX;
}
printf ("The sum is %d\n", sum);
```

The if Statement

- An example of an `if` statement:

```
if (sum > MAX)
{
   delta = sum - MAX;
}
printf ("The sum is %d\n", sum);
```

  - **First the condition is evaluated -- the value of `sum` is either greater than the value of `MAX`, or it is not**

The if Statement

- An example of an `if` statement:

```
if (sum > MAX)
{
   delta = sum - MAX;
}
printf ("The sum is %d\n", sum);
```

  - **First the condition is evaluated -- the value of `sum` is either greater than the value of `MAX`, or it is not**
  - **If the condition is true, the assignment statement is executed -- if it isn't, it is skipped.**

The if Statement

- An example of an `if` statement:

```
if (sum > MAX)
{
   delta = sum - MAX;
}
printf ("The sum is %d\n", sum);
```

  - **First the condition is evaluated -- the value of `sum` is either greater than the value of `MAX`, or it is not**
  - **If the condition is true, the assignment statement is executed -- if it isn't, it is skipped.**
  - **Either way, the call to `printf` is executed next**

Example: Age.c

- Write a C program that asks for user's age and checks if s/he is older than 21 years. If the user is older, then print a message that You are older.

The if Statement

- What do the following statements do?

The if Statement

- What do the following statements do?

```
if (top >= MAXIMUM)
   top = 0;
```

The if Statement

- What do the following statements do?

```
if (top >= MAXIMUM)
   top = 0;
```

Sets `top` to zero if the current value of `top` is greater than or equal to the value of `MAXIMUM`

The if Statement

- What do the following statements do?

```
if (top >= MAXIMUM)
   top = 0;
```

Sets `top` to zero if the current value of `top` is greater than or equal to the value of `MAXIMUM`

```
if (total != stock + warehouse)
   inventoryError = true;
```

The if Statement

- What do the following statements do?

```
if (top >= MAXIMUM)
   top = 0;
```

Sets `top` to zero if the current value of `top` is greater than or equal to the value of `MAXIMUM`

```
if (total != stock + warehouse)
   inventoryError = true;
```

Sets a flag to true if the value of `total` is not equal to the sum of `stock` and `warehouse`

The if Statement

- What do the following statements do?

```
if (top >= MAXIMUM)
   top = 0;
```

Sets `top` to zero if the current value of `top` is greater than or equal to the value of `MAXIMUM`

```
if (total != stock + warehouse)
   inventoryError = true;
```

Sets a flag to true if the value of `total` is not equal to the sum of `stock` and `warehouse`

- **The precedence of the arithmetic operators is higher than the precedence of the equality and relational operators**

Logical Operators

- C defines the following *logical operators*:

!	Logical NOT
&&	Logical AND
||	Logical OR

- Logical NOT is a unary operator (it operates on one operand)

- Logical AND and logical OR are binary operators (each operates on two operands)

Logical NOT

- The *logical NOT* operation is also called *logical negation* or *logical complement*
- If some condition `a` is true, then `!a` is false; if `a` is false, then `!a` is true
- Logical expressions can be shown using a *truth table*

a	**!a**
true	false
false	true

Logical AND and Logical OR

- The *logical AND* expression

```
a && b
```

  is true if both `a` and `b` are true, and false otherwise

- The *logical OR* expression

```
a || b
```

  is true if `a` or `b` or both are true, and false otherwise

Logical Operators

- Expressions that use logical operators can form complex conditions

Logical Operators

- Expressions that use logical operators can form complex conditions

```
if (total < MAX+5 && !found)
   printf ("Processing…");
```

Logical Operators

- Expressions that use logical operators can form complex conditions

```
if (total < MAX+5 && !found)
   printf ("Processing…");
```

  - **All logical operators have lower precedence than the relational operators**

Logical Operators

- Expressions that use logical operators can form complex conditions

```
if (total < MAX+5 && !found)
   printf ("Processing…");
```

  - **All logical operators have lower precedence than the relational operators**
  - **Logical NOT has higher precedence than logical AND and logical OR**

Logical Operators

- A truth table shows all possible true-false combinations of the terms
- Since `&&` and `||` each have two operands, there are four possible combinations of conditions `a` and `b`

`a`	`b`	`a && b`	`a || b`
true	true	true	true
true	false	false	true
false	true	false	true
false	false	false	false

Boolean Expressions

- Specific expressions can be evaluated using truth tables

`total < MAX`	`found`	`!found`	`total < MAX && !found`
false	false	true	false
false	true	false	false
true	false	true	true
true	true	false	false

Boolean Expressions in C

- C does not have a boolean data type.
- Therefore, C compares the values of variables and expressions against 0 (zero) to determine if they are true or false.
- If the value is 0 then the result is implicitly assumed to be false.
- If the value is different from 0 then the result is implicitly assumed to be true.
- C++ and Java have boolean data types.

The if-else Statement

- An *else clause* can be added to an `if` statement to make an *if-else statement*

The if-else Statement

- An *else clause* can be added to an `if` statement to make an *if-else statement*

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

The if-else Statement

- An *else clause* can be added to an `if` statement to make an *if-else statement*

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

  - **If the `condition` is true, `statement1` is executed; if the condition is false, `statement2` is executed**

The if-else Statement

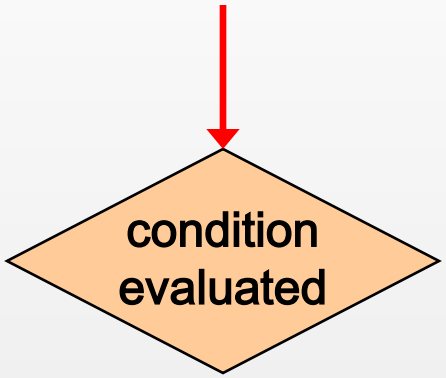
- An *else clause* can be added to an `if` statement to make an *if-else statement*

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

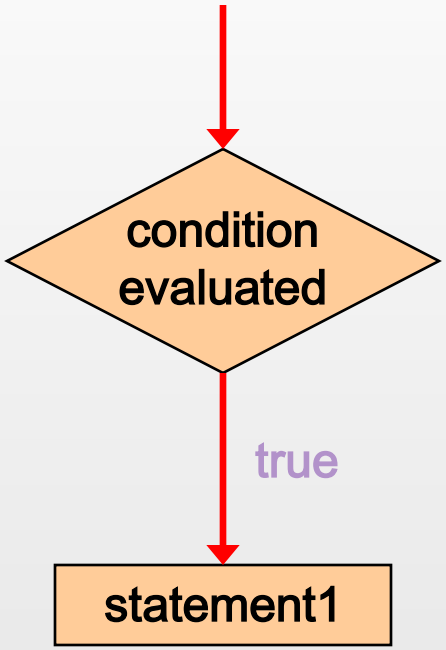
- **If the `condition` is true, `statement1` is executed; if the condition is false, `statement2` is executed**
- **One or the other will be executed, but not both**

Logic of an if-else statement

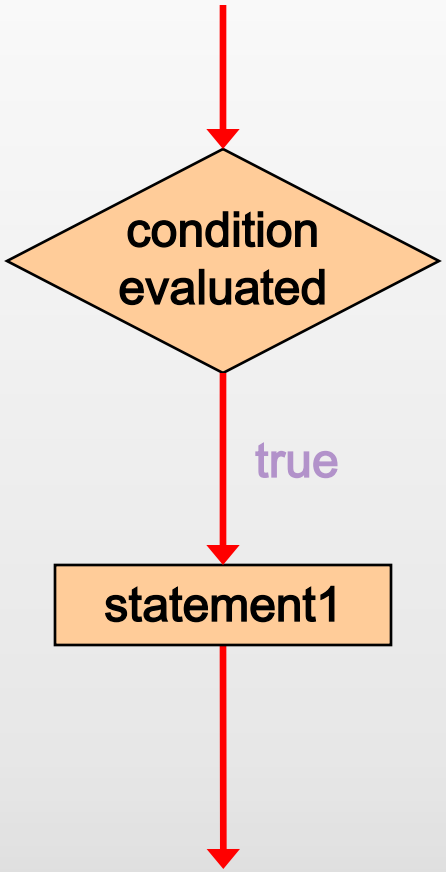
Logic of an if-else statement



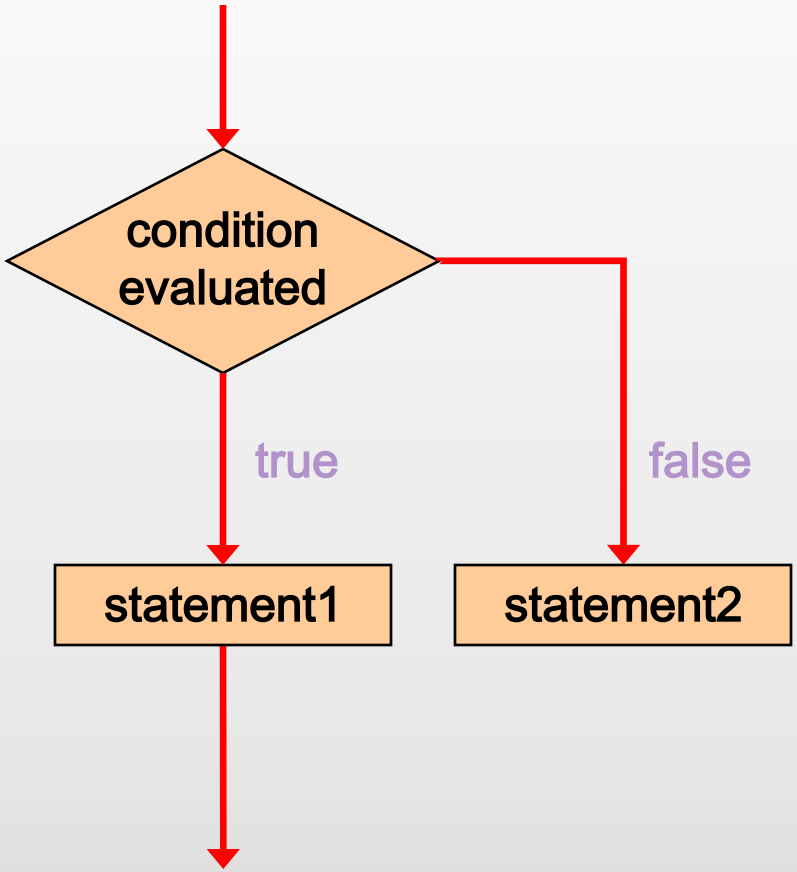
Logic of an if-else statement



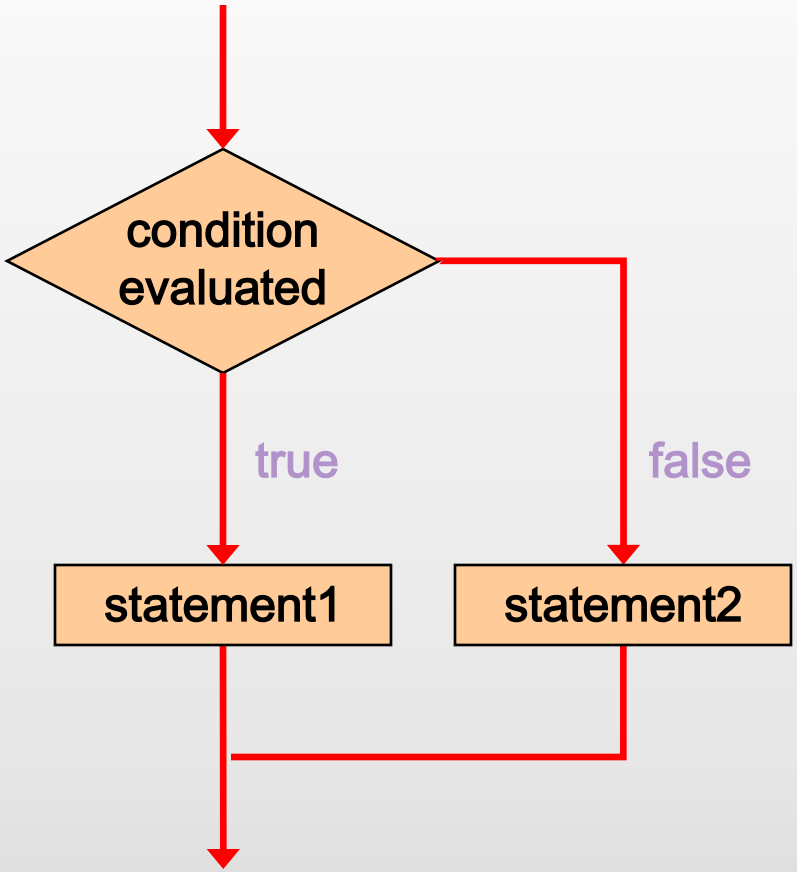
Logic of an if-else statement



Logic of an if-else statement



Logic of an if-else statement



Example

```
#include<stdio.h>
int main()
{
   int num;
   printf("Enter an integer number:");
   scanf("%d", &num);

   if(num<0)
   {
      printf("Number is negative.");
   }
}
```

```
#include<stdio.h>
int main()
{
   int num;
   printf("Enter an integer number:");
   scanf("%d", &num);

   if(num<0)
   {
      printf("Number is negative.");
   }
   printf("The number is positive");
}
```

Example

```
#include<stdio.h>
int main()
{
   int num;
   printf("Enter an integer number:");
   scanf("%d", &num);

   if(num<0)
   {
      printf("Number is negative.");
   }
}
```

```
#include<stdio.h>
int main()
{
   int num;
   printf("Enter an integer number:");
   scanf("%d", &num);

   if(num<0)
   {
      printf("Number is negative.");
   }
   printf("The number is positive");
}
```

This line will always execute. Even if The number is not positive.

Sample Program

```
#include<stdio.h>
int main()
{
   int num;
   printf("Enter an integer number:");
   scanf("%d", &num);

   if(num<0)
   {
       printf("The number is negative.");
   }
   else
   {
       printf("The number is positive");
   }
}
```

Example: Age.c

- Write a C program that asks for user's age and checks if s/he is older than 21 years. If the user is older, then print a message that You are older. And if not then print you are not older.

Example: Wages.c

- Write a C program that calculates weekly wages for hourly employees.
- Regular hours 0-40 are paid at $10/hours.
- Overtime (> 40 hours per week) is paid at 150$/extra hours.