C++ Control Structures

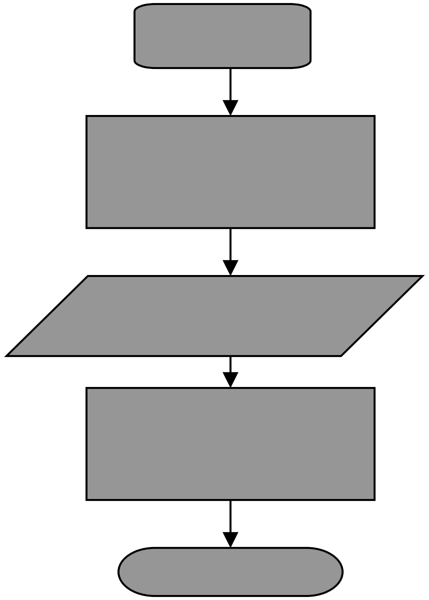
Part I - Decisions

Control Structures

- Sequential
- Decision
- Case
- Loop

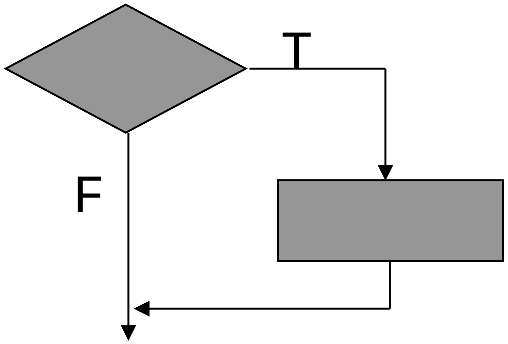
Sequential

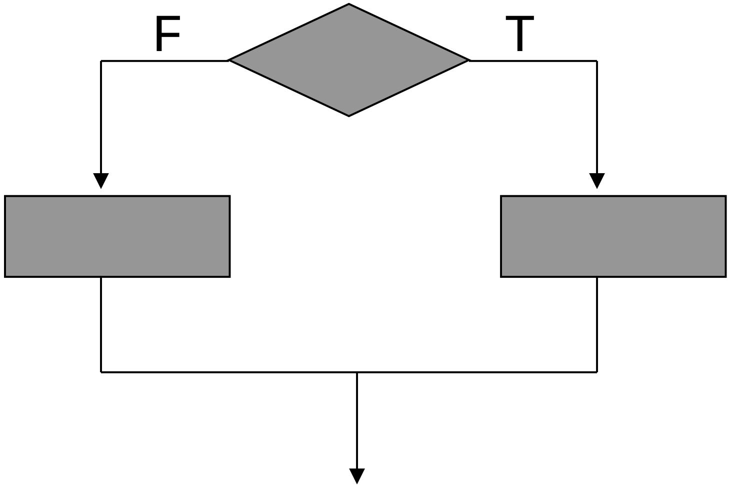
- Each command executed one after the other.



Decision

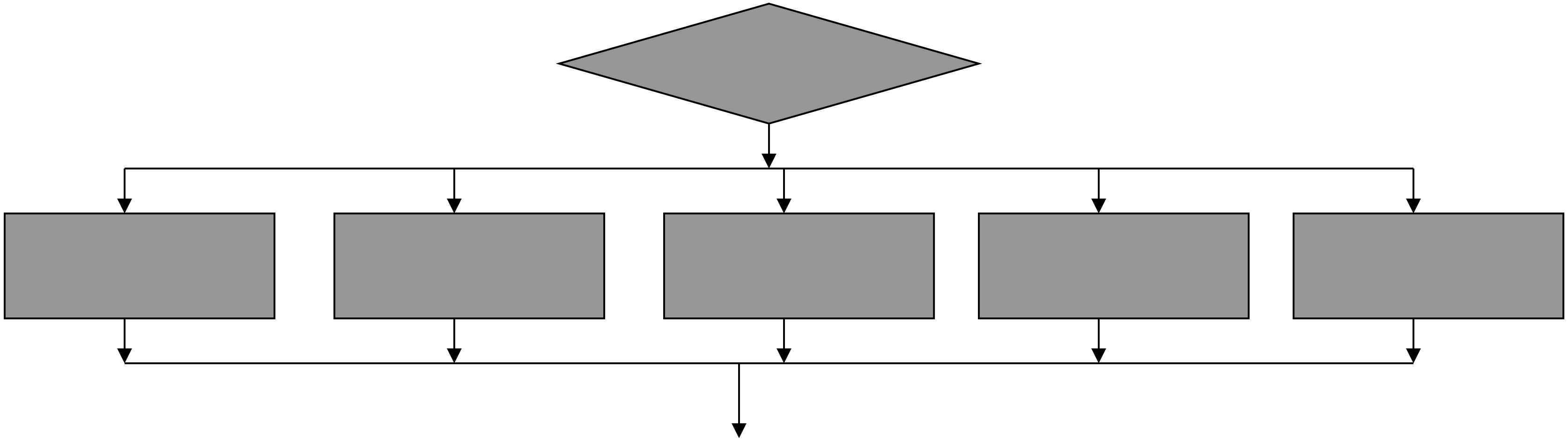
- An expression is evaluated to true or false. Depending on the value of the expression one of two paths is chosen.





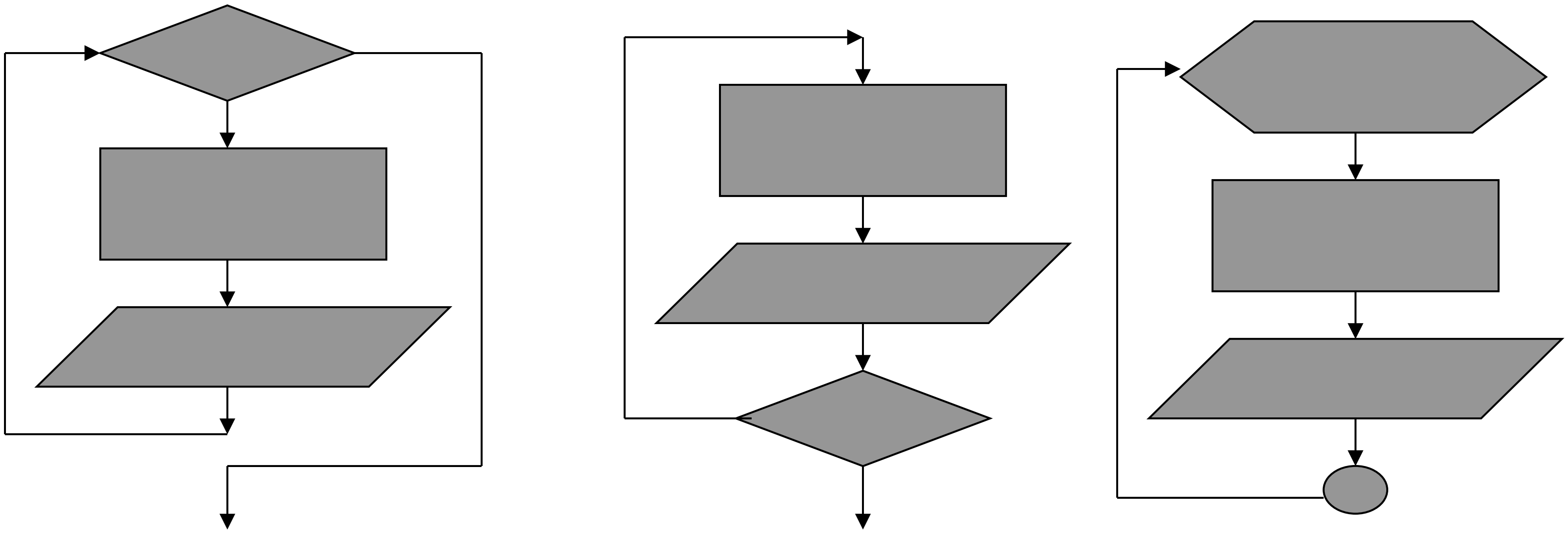
Case

- Based on the value of a variable, a switch statement can choose from several courses of action.



Repetition - Loops

- A set of commands need to executed multiple times.



Logical (Boolean) Expression

- An expression that evaluates to true or false
- Relational Operators:

  <, <=, >, >=, ==, !=
- Logical Operators:

  ! – NOT

  && – AND

  || – OR

Logical Operators

! Takes one logical operand.
Returns the opposite of the logical value

&& Takes two logical operands.
Returns true if both operands are true, false otherwise

|| Takes two logical operands.
Returns true if either operand is true, false otherwise

Logical Operators

A	!A
true	
false	

Logical Operators

A	!A
true	false
false	true

Logical Operators

A	B	A && B
true	true	
true	false	
false	true	
false	false	

Logical Operators

A	B	A && B
true	true	true
true	false	false
false	true	false
false	false	false

Logical Operators

A	B	A || B
true	true	
true	false	
false	true	
false	false	

Logical Operators

`A`	`B`	`A || B`
`true`	**`true`**	**`true`**
`true`	**`false`**	**`true`**
`false`	**`true`**	**`true`**
`false`	**`false`**	**`false`**

Logical Operators

Certain acceptable mathematical or English expressions are not valid C++ statements

Examples:

0 < x < 8 vs. 0 < x && x < 8

x == 1 || 5 vs. x == 1 || x == 5

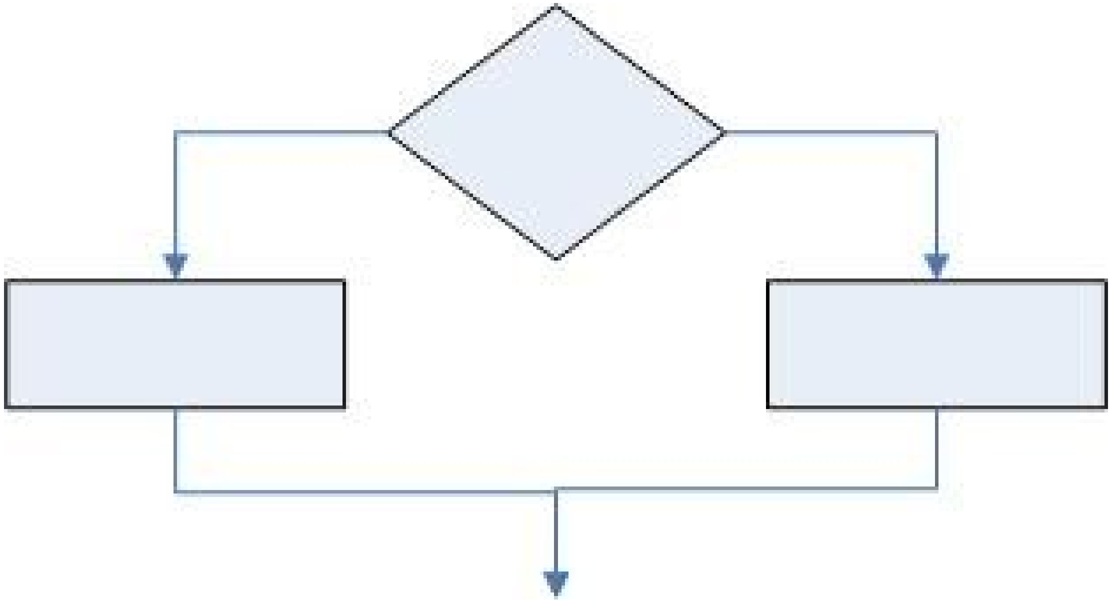
Order of Operations

! then && then ||

Check out the precedence chart on pages 53-54 of your text to see how these fit in among the other C++ operators

if-else statements

- Allows program to choose between two paths.



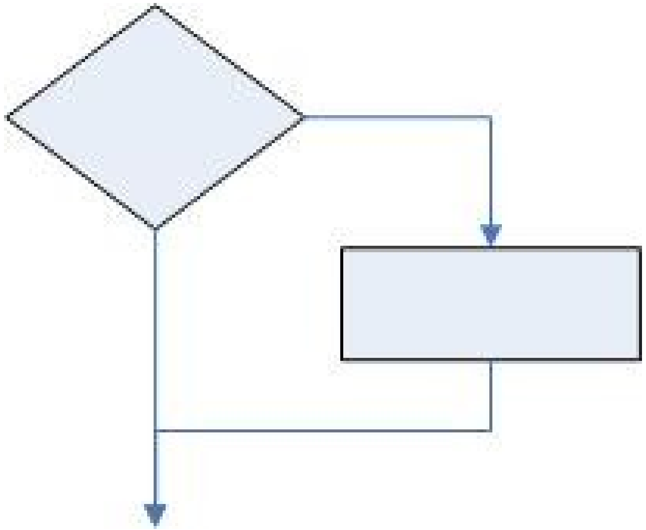
if-else statements

Syntax:

```
if ( logical expression ) {
    true side actions
}
else {
    false side actions
}
```

if-else statements

- else is not required



- Braces are not required if only one action occurs on the true side of the decision.

if-else statements

Nesting – placing decisions on the true or false sides of other decisions

If there are no braces, each else will be paired with the most recent, unmatched if.

However… the Google style guide states that we should use braces any time we have an else or nest decisions

Ternary Conditional Operator

- Most operators that we have seen to this point have been binary operators (+,-,*,/,%,=, ==,!=,<, >,<=,>=,<<,>>) which means that they take two operands.
- The conditional operator (?:) takes three operands.

Ternary Conditional Operator

This operator works like an if/else statement

Logical	True Action		False Action
Expression ?	or Value	:	or Value

The true and false actions / values must evaluate to the same type.

Ternary Conditional Operator

For example, the conditional statement below is equivalent to the if–else statement that follows

```
answer == ‘Y’ ? cout << “Good” :   cout << “*#!”;

if ( answer == ‘Y’ ) {
  cout << “Good”;
} else {
  cout << “*#!”;
}
```

Ternary Conditional Operator

Why do we need the conditional operator when we already have an if–else structure?

The conditional operator can be used inside of other statements – an if statement can not.

Note – this operator has a very low precedence (see pp53-54)

Examples

```
cout << (Answer == ‘Y’ ? “Good” : “*#!”);


ticketPrice = age < 4 ? 0 : 7;
```

is equivalent to

```
if ( age < 4 ) {
  ticketPrice = 0;
} else {
  ticketPrice = 7;
}
```