

## If Control Construct

A mechanism for deciding whether an action should be taken

JPC and JWD © 2002 McGraw-Hill, Inc.

## Boolean Algebra

- ◆ Logical expressions have the one of two values - true or false
  - A rectangle has three sides
  - The instructor has a pleasant smile
- ◆ The branch of mathematics is called Boolean algebra
  - Developed by the British mathematician George Boole in the 19th century
- ◆ Three key logical operators
  - And
  - Or
  - Not

# Boolean Algebra

## ◆ Truth tables

- Lists all combinations of operand values and the result of the operation for each combination

## ◆ Example

P	Q	P and Q
False	False	False
False	True	False
True	False	False
True	True	True

# Boolean Algebra

## ◆ Or truth table

P	Q	P or Q
False	False	False
False	True	True
True	False	True
True	True	True

# Boolean Algebra

- ◆ Not truth table

P	not P
False	True
True	False

# Boolean Algebra

- ◆ Can create complex logical expressions by combining simple logical expressions
- ◆ Example
  - not (P and Q)
- ◆ A truth table can be used to determine when a logical expression is true

P	Q	P and Q	not (P and Q)
False	False	False	True
False	True	False	True
True	False	False	True
True	True	True	False

## A Boolean Type

- ◆ C++ contains a type named `bool`
- ◆ Type `bool` has two symbolic constants
  - `true`
  - `false`
- ◆ Boolean operators
  - The and operator is `&&`
  - The or operator is `||`
  - The not operator is `!`
- ◆ Warning
  - `&` and `|` are also operators so be careful what you type

## A Boolean Type

- ◆ Example logical expressions

```
bool P = true;
bool Q = false;
bool R = true;
bool S = (P && Q);
bool T = ((!Q) || R);
bool U = !(R && (!Q));
```

# Relational Operators

## ◆ Equality operators

- ==
- !=

## ◆ Examples

- `int i = 32;`
- `int k = 45;`
- `bool q = (i == k);`
- `bool r = (i != k);`

# Relational Operators

## ◆ Ordering operators

- <
- >
- >=
- <=

## ◆ Examples

- `int i = 5;`
- `int k = 12;`
- `bool p = (i < 10);`
- `bool q = (k > i);`
- `bool r = (i >= k);`
- `bool s = (k <= 12);`

## Operator Precedence Revisited

◆ Precedence of operators (from highest to lowest)

- Parentheses
- Unary operators
- Multiplicative operators
- Additive operators
- Relational ordering
- Relational equality
- Logical and
- Logical or
- Assignment

## Operator Precedence Revisited

◆ Consider

```
5 * 15 + 4 == 13 && 12 < 19 || !false == 5 < 24
```

## Operator Precedence Revisited

- ◆ Consider

```
5 * 15 + 4 == 13 && 12 < 19 || !false == 5 < 24
```

- ◆ Yuck! Do not write expressions like this!

## Operator Precedence Revisited

- ◆ Consider

```
5 * 15 + 4 == 13 && 12 < 19 || !false == 5 < 24
```

- ◆ However, for your information it is equivalent to

```
((((5 * 15) + 4) == 13) && (12 < 19))
```

```
||
```

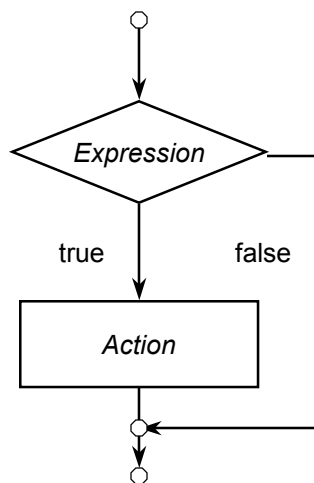
```
((!false) == (5 < 24))
```

# Conditional Constructs

- ◆ Provide
  - Ability to control whether a statement list is executed
- ◆ Two constructs
  - If statement
    - ◆ if
    - ◆ if-else
    - ◆ if-else-ef
  - Switch statement
    - ◆ Left for reading

# The Basic If Statement

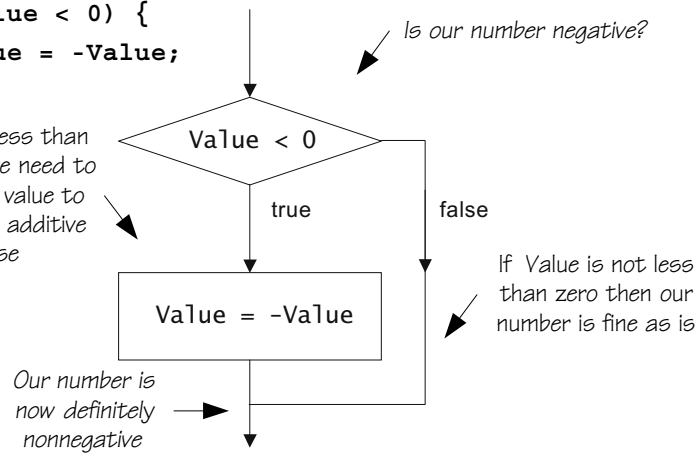
- ◆ Syntax
  - if* (*Expression*)
  - Action*
- ◆ If the *Expression* is true then execute *Action*
- ◆ *Action* is either a single statement or a group of statements within braces



## Example

```
if (Value < 0) {  
    Value = -Value;  
}
```

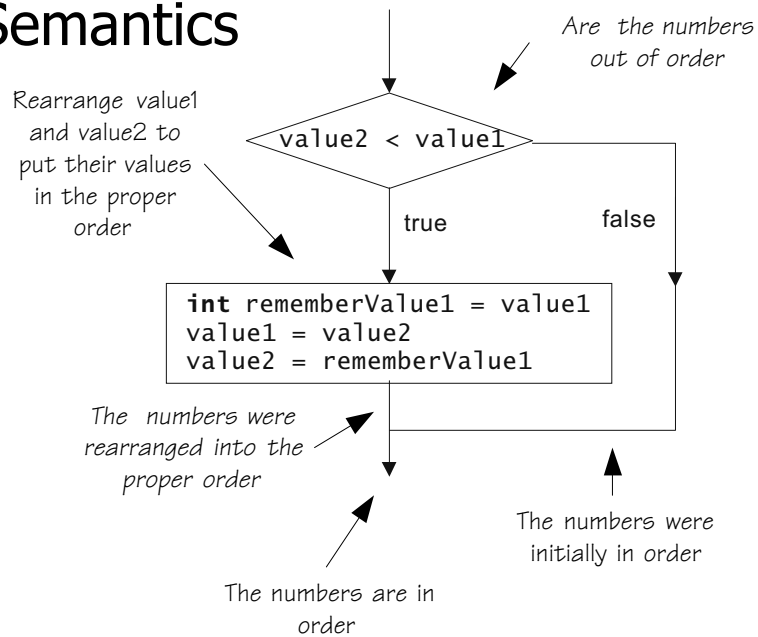
If Value is less than zero then we need to update its value to that of its additive inverse



## Sorting Two Numbers

```
cout << "Enter two integers: ";  
int Value1;  
int Value2;  
cin >> Value1 >> Value2;  
if (Value1 > Value2) {  
    int RememberValue1 = Value1;  
    Value1 = Value2;  
    Value2 = RememberValue1;  
}  
cout << "The input in sorted order: "  
    << Value1 << " " << Value2 << endl;
```

## Semantics



## What is the Output?

```
int m = 5;  
int n = 10;  
  
if (m < n)  
    ++m;  
    ++n;  
  
cout << " m = " << m << " n = " << n << endl;
```

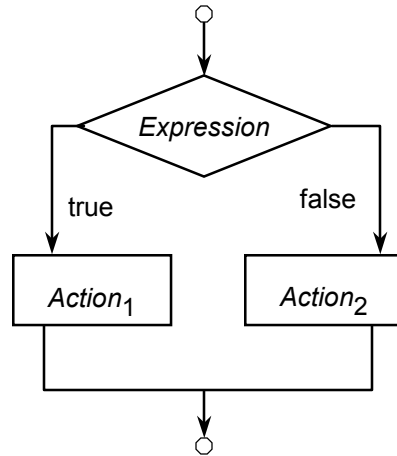
# The If-Else Statement

## ◆ Syntax

```
if (Expression)  
    Action1  
else  
    Action2
```

## ◆ If *Expression* is true then execute *Action*<sub>1</sub> otherwise execute *Action*<sub>2</sub>

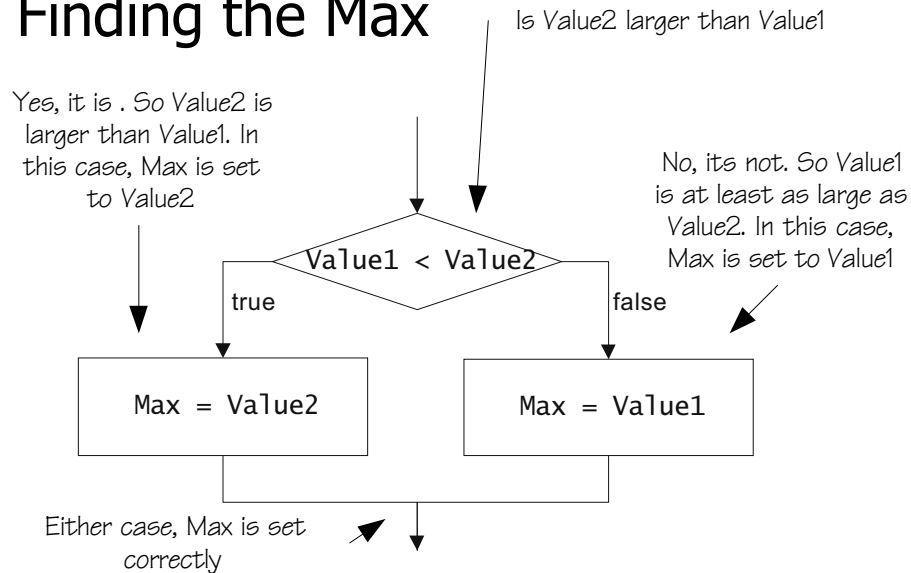
```
if (v == 0) {  
    cout << "v is 0";  
}  
else {  
    cout << "v is not 0";  
}
```



# Finding the Max

```
cout << "Enter two integers: ";  
int Value1;  
int Value2;  
cin >> Value1 >> Value2;  
int Max;  
if (Value1 < Value2) {  
    Max = Value2;  
}  
else {  
    Max = Value1;  
}  
cout << "Maximum of inputs is: " << Max << endl;
```

## Finding the Max



## Selection

- ◆ It is often the case that depending upon the value of an expression we want to perform a particular action
- ◆ Two major ways of accomplishing this choice
  - if-else-if statement
    - ◆ if-else statements "glued" together
  - Switch statement
    - ◆ An advanced construct

## An If-Else-If Statement

```
if ( nbr < 0 ){
    cout << nbr << " is negative" << endl;
}
else if ( nbr > 0 ) {
    cout << nbr << " is positive" << endl;
}
else {
    cout << nbr << " is zero" << endl;
}
```

## A Switch Statement

```
switch (ch) {
    case 'a': case 'A':
    case 'e': case 'E':
    case 'i': case 'I':
    case 'o': case 'O':
    case 'u': case 'U':
        cout << ch << " is a vowel" << endl;
        break;
    default:
        cout << ch << " is not a vowel" << endl;
}
```

```
cout << "Enter simple expression: ";
int Left;
int Right;
char Operator;
cin >> Left >> Operator >> Right;
cout << Left << " " << Operator << " " << Right
    << " = ";
switch (Operator) {
    case '+' : cout << Left + Right << endl; break;
    case '-' : cout << Left - Right << endl; break;
    case '*' : cout << Left * Right << endl; break;
    case '/' : cout << Left / Right << endl; break;
    default: cout << "Illegal operation" << endl;
}
```