Making Decisions using if-else statements

Course Code: CSC1102 &1103 Course Title: Introduction to Programming

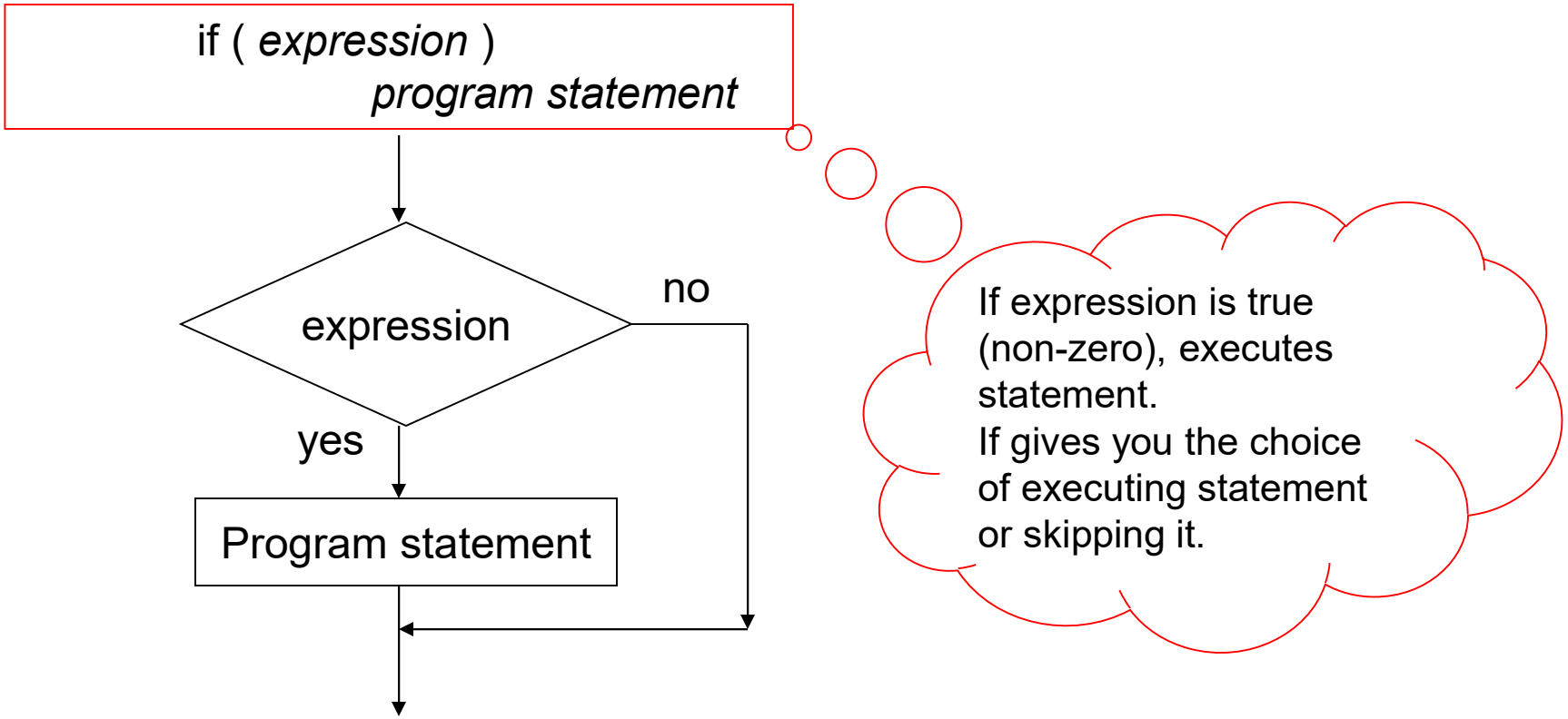
Dept. of Computer Science
Faculty of Science and Technology

Lecturer No:	**3**	**Week No:**	**2 (2X1.5 hrs)**	**Semester:**	
Lecturer:	*Name & email*				

Lecture 3: Outline

- **Making Decisions**
- The `if` Statement
  - The `if-else` Construct
  - Logical Operators
  - Boolean Variables
  - Nested `if` Statements
  - The `else if` Construct
  - The `switch` Statement
  - The Conditional Operator
- **Character Input/Output**

The `if` statement

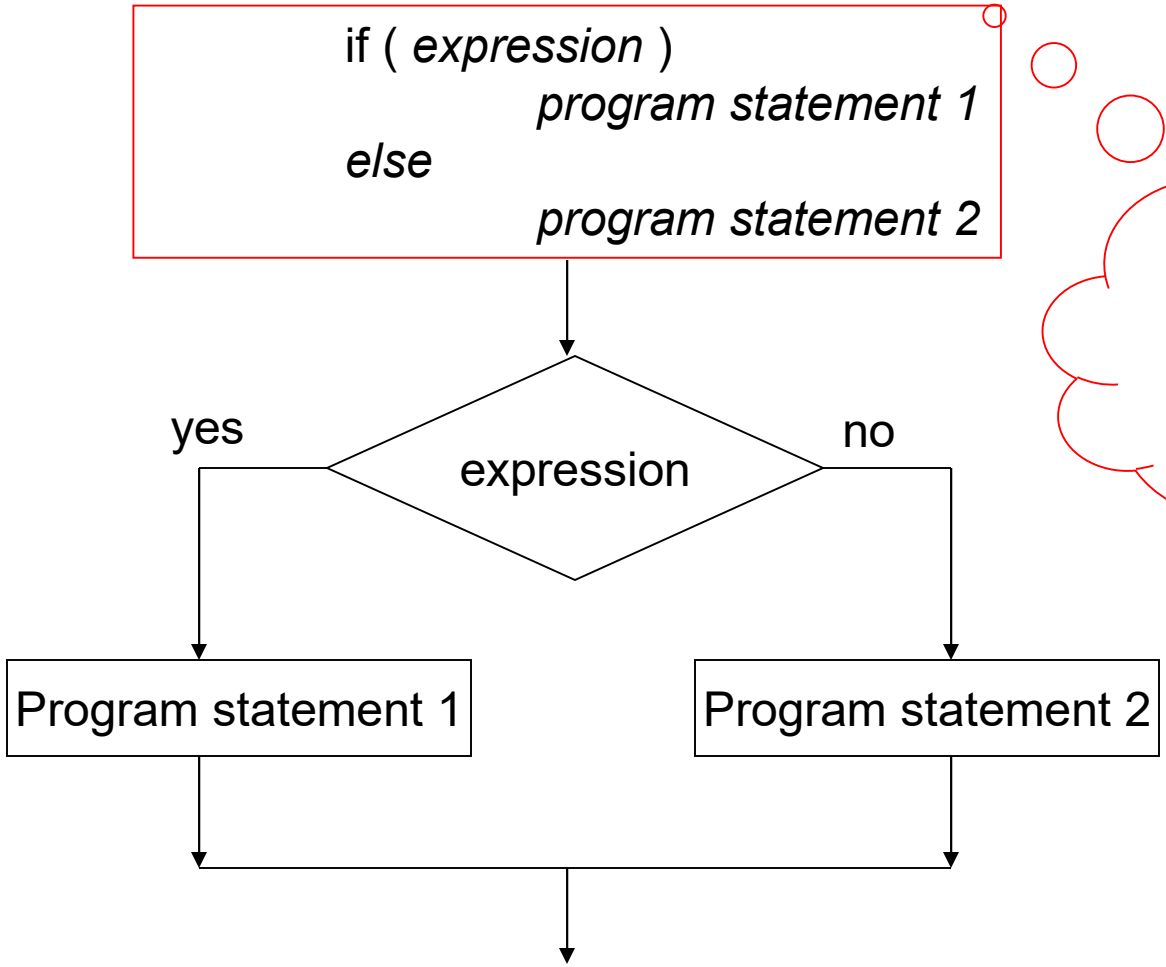


Example - if

```
// Program to calculate the absolute value of an integer
int main (void)
{
    int number;
    cout<<"Type in your number: "<<endl;
    cin>>number;
    if ( number < 0 )
        number = -number;
    cout<<"The absolute value is "<<number<<endl;
    return 0;
}
```

The `if-else` statement

```
if ( expression )
        program statement 1
else
        program statement 2
```



if-else statement: enables you to choose between two statements

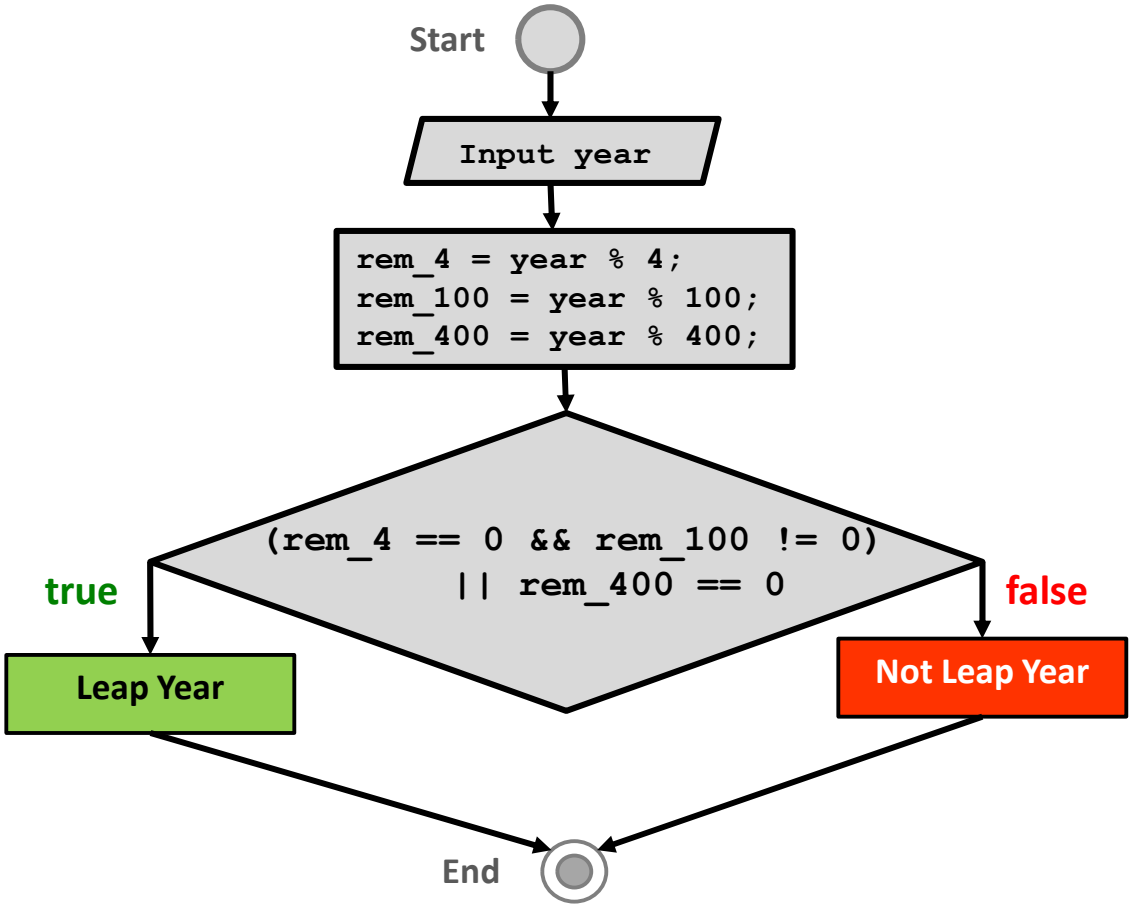
Example: `if-else`

```
// Program to determine if a number is even or odd
int main ()
{
    int number_to_test, remainder;
    cout<<"Enter your number to be tested: "<<endl;
    cin>>number_to_tes;

    remainder = number_to_test % 2;

    if ( remainder == 0 )
        cout<<"The number is even"<<endl;
    else
        cout<<"The number is odd"<<endl;
    return 0;
}
```

Example: compound relational test

```
/* Program to determine if a number is even or odd
and also the number cannot be negative */
int main ()
{
    int number_to_test, remainder;
    cout<<"Enter your number to be tested: "<<endl;
    cin>>number_to_tes;

    remainder = number_to_test % 2;

    if (number_to_test >= 0 && remainder == 0 )
        cout<<"The number is even"<<endl;
    else
        cout<<"The number is odd"<<endl;
    return 0;
}
```

Flowchart to determine if a year is a leap year



Example: compound relational test

```
// Program to determine if a year is a leap year or not
int main ()
{
    int year, rem_4, rem_100, rem_400;
    cout<<"Enter the year to be tested: "<<endl;
    cin>>year;
    rem_4 = year % 4;
    rem_100 = year % 100;
    rem_400 = year % 400;

    if ( (rem_4 == 0 && rem_100 != 0) || rem_400 == 0 )
        cout<<"It's a leap year."<<endl;
    else
        cout<<"It's not a leap year."<<endl;

    return 0;
}
```

Logical operators

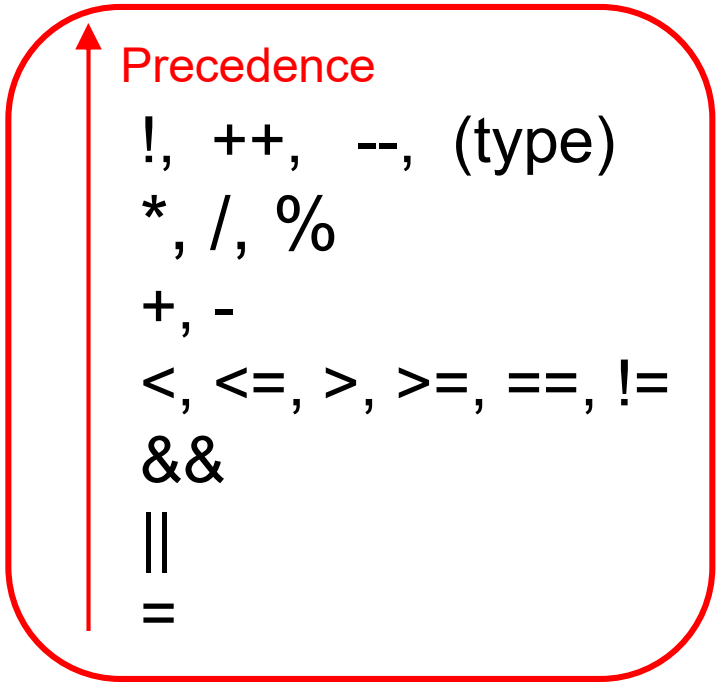
Operator	Symbol	Meaning
AND	`&&`	X && y is true if BOTH x and y are true
OR	`||`	X || y is true if at least one of x and y is true
NOT	!	!x is true if x is false

Logical values as operands or in tests: true = non-zero, false=zero

Logical values returned as results of expressions: true = 1, false=zero

Example: `5 || 0` is `1`

Precedence of operators



Example for operator precedence:

```
a > b && b > c || b > d
```

Is equivalent to:

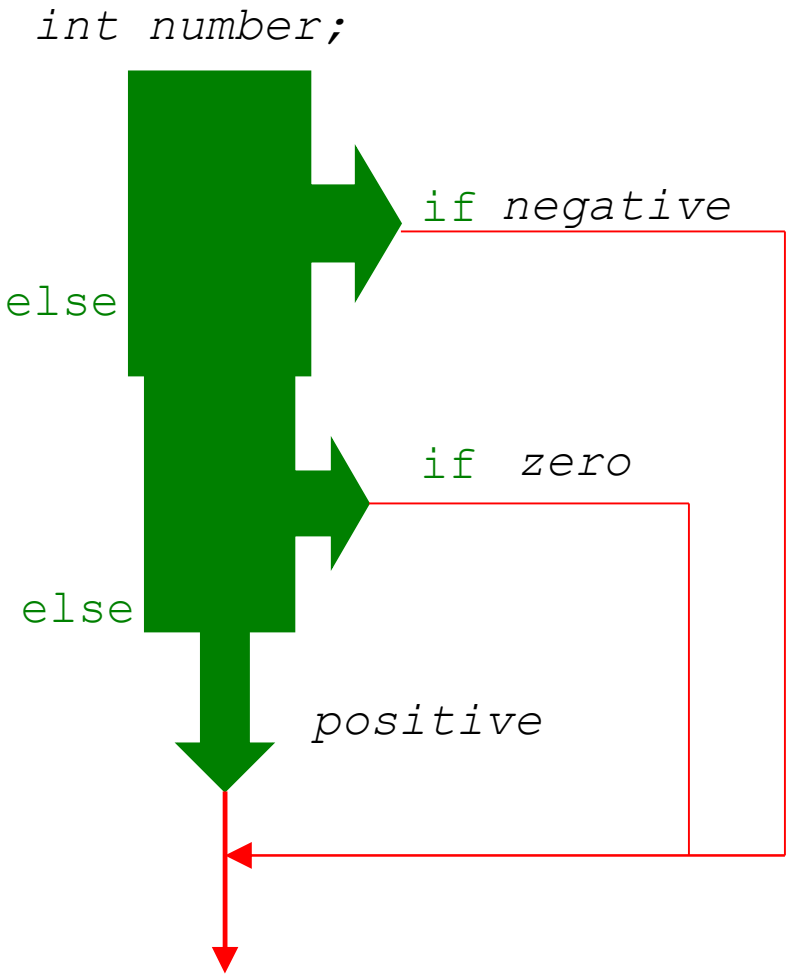
```
((a > b) && (b > c)) || (b > d)
```

Nested if statements

```
if (condition)
{
      if (condition)
      else
}
else
{
      if (condition)
      else
}
```

```
void main()
{
      int a,b,c;

      cout << "\nEnter value of A : ";
      cin >> a;

      cout << "\nEnter value of B : ";
      cin >> b;

      cout << "\nEnter value of C++ : ";
      cin >> c;

      if (a>b)
       {
              if (a>c)
                cout << "\n\nA is Greatest";
               else
                 cout << "\n\nC is Greatest";
        }
      else
        {
               if (b>c)
                  cout << "\n\nB is Greatest";
                else
                  cout << "\n\nC is Greatest";
         }
}
```

Multiple choices – `else-if`




```
if ( expression 1)
    program statement 1
else if ( expression 2)
    program statement 2
else
    program statement 3
```

Program style: this unindented formatting improves the readability of the statement and makes it clearer that a three-way decision is being made.

Example – multiple choices

```
/* Program to evaluate simple expressions of the form
number operator number */
int main () {
    float value1, value2;
    char operator;
    cout<<"Type in your expression"<<endl;
    cin>>value1>>operator>>value2;
    if ( operator == '+' )
        cout<<value1 + value2<<endl;
    else if ( operator == '-' )
        cout<<value1 - value2<<endl;
    else if ( operator == '*' )
        cout<<value1 * value2<<endl;
    else if ( operator == '/' )
        cout<<value1 / value2<<endl;
    else cout<<"Unknown operator.";
    return 0;
}
```

The `switch` statement

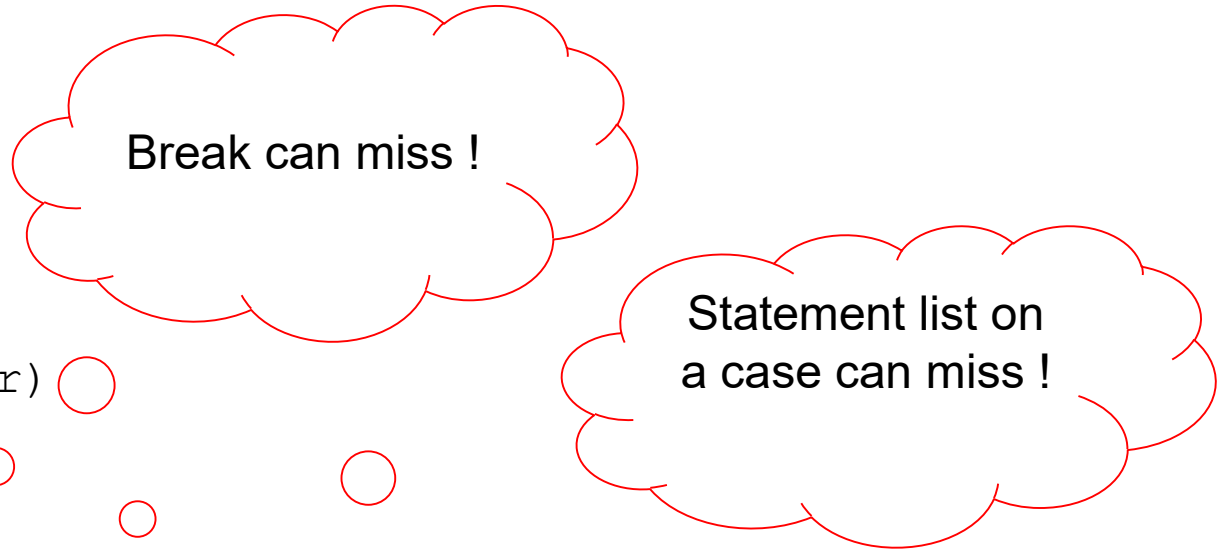
```
switch ( expression )
{
    case value1:
        program statement
        ...
        break;
    case value2:
        program statement
        ...
        break;
     case valueN:
        program statement
        ...
        break;
    default:
        program statement
        ...
        break;
}
```

The *expression* is successively compared against the values *value1*, *value2, ...*, *valuen*. If a case is found whose value is equal to the value of *expression*, the program statements that follow the case are executed.

The switch test expression must be one with an integer value (including type char) (No float !).

The case values must be integer-type constants or integer constant expressions (You can't use a variable for a case label !)

The `switch` statement (cont)




```
switch (operator)
{
    ...
    case '*':
    case 'x':
        printf ("%.2f\n", value1 * value2);
        break;
    ...
}
```

Example - switch

```
char choice;
cin >> choice
switch(choice) {
    case 'Y' :
        cout << "Yes";
        break;
    case 'M' :
        cout << "Maybe";
        break;
    case 'N' :
        cout << "No";
        break;
    default:
        cout << "Invalid response";
}
```

The conditional operator

condition ? *expression1* : *expression2*

condition is an expression that is evaluated first.
If the result of the evaluation of *condition* is TRUE (nonzero), then *expression1* is evaluated and the result of the evaluation becomes the result of the operation.
If *condition* is FALSE (zero), then *expression2* is evaluated and its result becomes the result of the operation

```
maxValue = ( a > b ) ? a : b;
```

Equivalent to:

```
if ( a > b )
        maxValue = a;
else
        maxValue = b;
```