Lecture 3

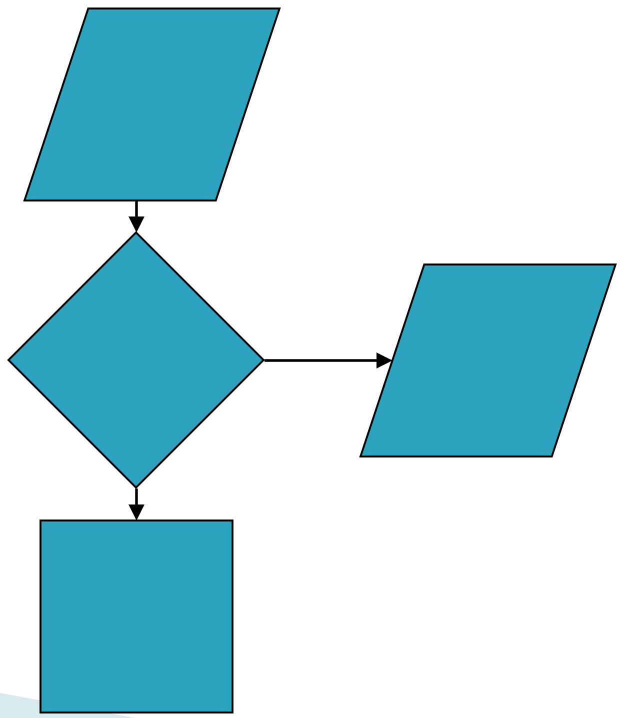
Decision Making Statements

Decision Making In Computers

- A circuit quite simply allows one out of two choices to be made depending on its inputs
- When decisions are made in a computer program, they are simply the result of a computation in which the final result is either **TRUE** or **FALSE**
- The value zero (0) is considered to be FALSE by C++. Any positive or negative value is considered to be TRUE

Programming & Decisions

- Practically all computer programs, when modeled with a flowchart, demonstrate that branching occurs within their algorithms.



Decision Making in C++

1. if statement
2. switch statement
3. ? conditional operator statement
4. goto statement

Using Relational Operators

- Relational operators provide the tools with which programs make decisions with true and false evaluations

==	equal to NOTE: this is two equals symbols next to each other, not to be confused with the assignment operator, =
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
!=	not equal to

Using Logical Operators

- When complex decisions must be coded into an algorithm, it may be necessary to "chain together" a few relational expressions (that use relational operators)
- This is done with **logical operators** (also called **Boolean operators**.)

&&	is the logical AND operator
||	is the logical OR operator
!	is the logical NOT operator

Truth Tables

- Use this **truth table** to determine the results of the logical operators. In this table, 1 represents TRUE and 0 represents FALSE.
- Note that the ! symbol (the logical NOT operator) changes a TRUE to a FALSE.

AND				**OR**				**NOT**	
A	B	A && B		A	B	A || B		A	!A
0	0	0		0	0	0		0	1
0	1	0		0	1	1		1	0
1	0	0		1	0	1			
1	1	1		1	1	1			

Compare and Branch

- A program can instruct a computer to compare two items and do something based on a match or mismatch which, in turn, redirect the sequence of programming instructions.
  - There are two forms:
  - IF–THEN
  - IF–THEN–ELSE

Levels of Complexity for if

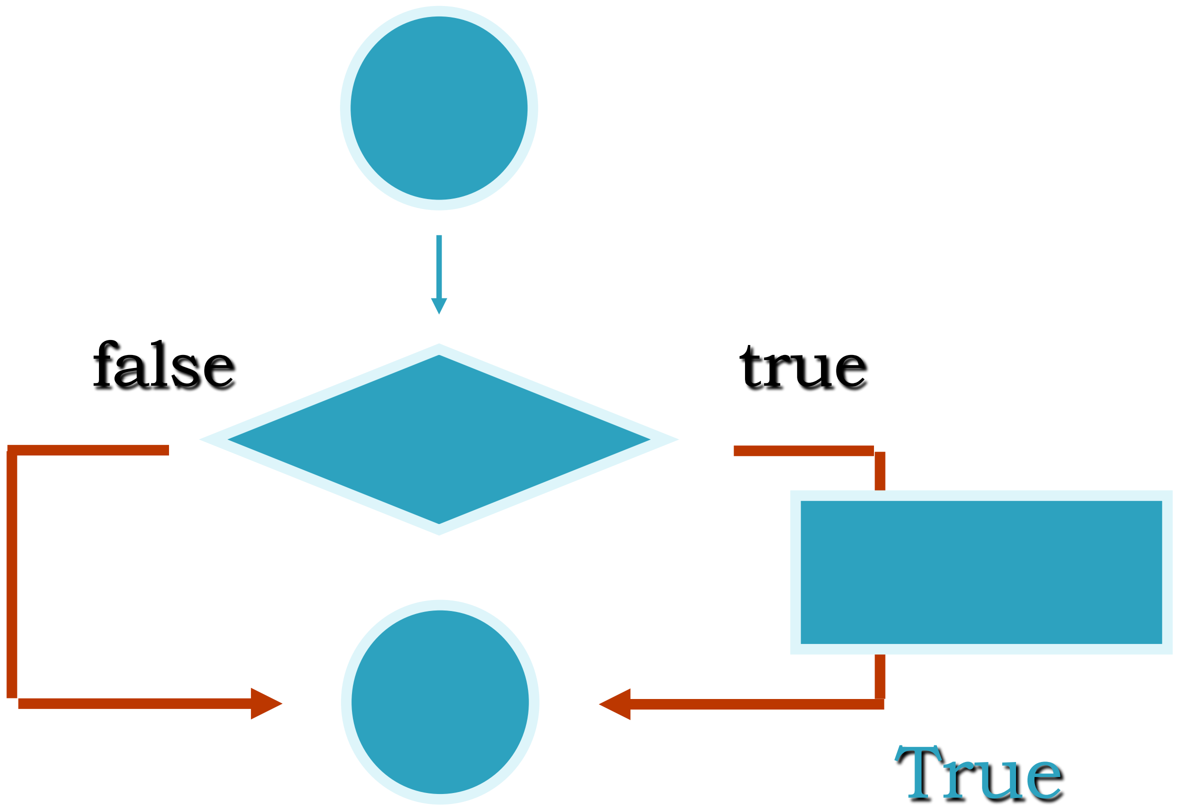
- Simple if statement
- if … else statement
- Nested if … else statement
- else..if ladder

IF-THEN

Entry

Test condition p

Exit



True statement a

Use the “IF” structure

- Practically all computer languages have some sort of if structure. In C++, the if structure is a one-way selection structure:

```
if (number == 3)
{
        Printf("The value of number is 3“);;
}
```

- IF structures use a **control expression** to determine if the code in the braces is to be executed or not

Compound Conditionals

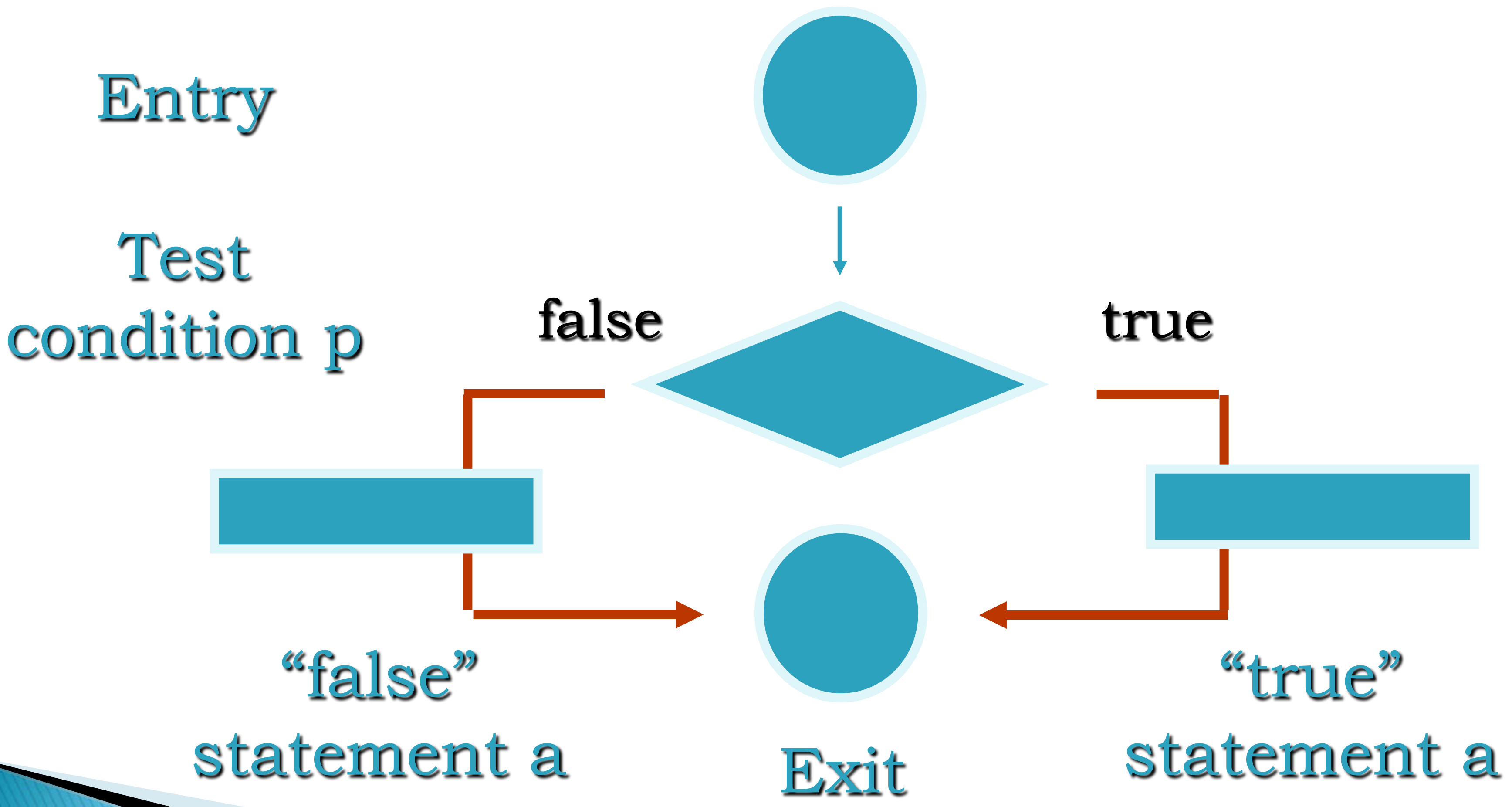
- You must use the AND operator (&&) to form a compound relational expression:

```
if (0 < number && number < 10)
{
      printf(‘’number  is greater than 0 
  but less than 10”);
}
```

Coding IF Structures

- Place a semicolon at the end of each statement within the braces, which can contain many executable statements
- Use curly braces around the body of an IF structure even if there is only one statement

IF…ELSE


Entry
Test condition p
false
true
“false” statement a
Exit
“true” statement a

General Form

```
if (test expression)
{
 True-block statements;
}
else
{
 False-block statements;
}

next statement;
```

The If … Else statement

- Two-way selection structure since either the block of code after the "if" part will be executed or the block of code after the "else" part will be executed

- The “If" part is executed if the control expression is TRUE while the "else" part is executed if the "if" part is FALSE, guaranteeing one part of the expression to be executed or the other

if… else if Ladder: General Form

```
if (condition 1)
  statement 1;
else if (condition 2)
  statement 2;
else if (condition 3)
  statement 3;
else if (condition n)
  statement n;
else
  default statement;
statement x;
```

if/else if statement

- As soon as a true condition is found, the statement associated with it is executed and control is transferred to the statement after the ladder
- The else clause is optional just as it is with an if statement.

```
if (number > 10)
        {
        printf(" number is greater than 10." );
        }
else if (number <= 9 && number >= 6)
        {
           printf("number" is greater than 5
             and less than 10.");
             }
            else
             {
           printf("number must be less than 6." );
         }
number = number + 1;
```

Nested If Statements

- If structures and if/else statements can be nested within one another in order to model complex decision structures.
  - Use the braces and semicolons properly when coding such structures.

General Form

```
if (test condition 1)
  { // true-block1 statements
         if (test condition 2)
         {
         true-block2 statements;
                     }
         else
          {
          false-block2 statements;
                     }
    }
else
  {
  false-block1 statements;
   }
```

Conditional Operator

- General form:

  conditional expression ? exp-1:exp-2 ;

  - The conditional expression is evaluated first. If the result is non-zero, exp-1 is evaluated and is returned as the value of the conditional expression. Otherwise, exp-2 is evaluated and its value is returned

  f=(c = = 'y') ? 1: 0;

GOTO Statement

- Unconditional branching

```
GOTO label;
…
label: statement;
```

- Not recommended but may be used occasionally