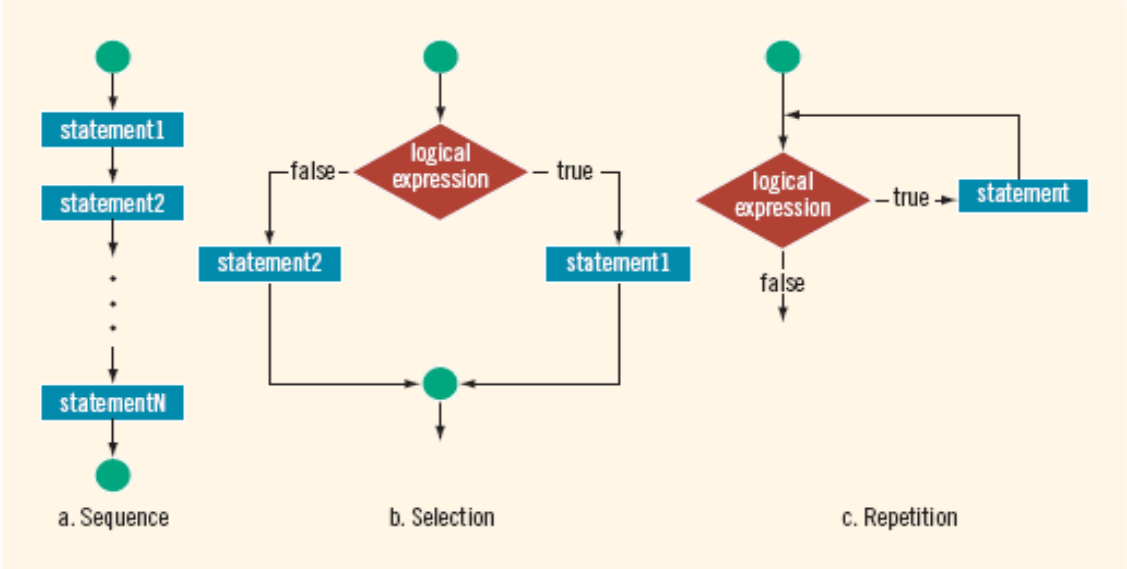
Chapter 4 – Making Decisions

Three ways a computer processes statements:

- Sequentially
- Selection
- Repetition

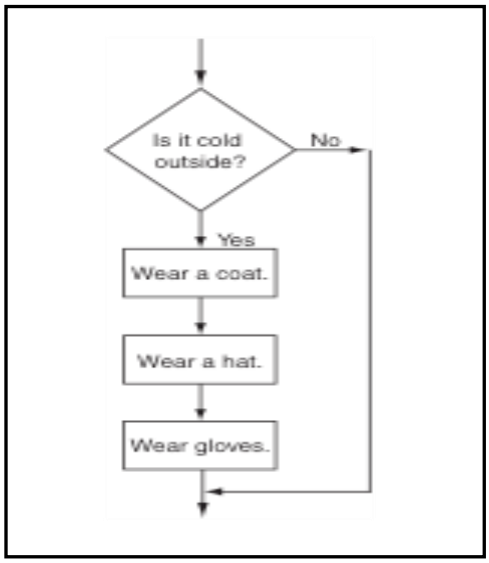


Straight-line code

- So far all of our programs have followed this basic format:
  - Input some values
  - Do some computations
  - Output the results
- The statements are executed in a sequence, first to last.

Decisions

- Sometimes we want to be able to decide NOT to execute certain statements:



Relational Expressions

- Making decisions require being able to ask "Yes" or "No" questions.
- Relational expressions evaluate to true or false.
- Also called
  - logical expressions
  - conditional expressions
  - boolean expressions

Relational Expressions

- Boolean literals have two values :
  - true
  - false

Boolean variables:

```
bool isPositive;
bool found;

isPositive = true;    // isPositive evaluates to  true
found = false;        // found evaluates to  false
```

Relational Operators

- Express conditions
- Make comparisons to make decisions
- **Logical** (**Boolean**) **expression**: has a value of either true or false
- **Relational operator**: allows comparisons in a program
- All Relational operators are binary
  - Binary operators require two operands
  - Result of comparison is true or false
- Relational operators used with integral and floating-point data types

Thus, Binary operators that are used to compare numbers are :

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

Expression	Meaning	Value
8 < 15	8 less than 15	True
6 != 6	6 not equal to 6	False
2.5 > 5.8	2.5 greater than 5.8	False
5.9 <= 7.5	5.9 less than or equal to 7.5	True

Comparing Characters

- Expression using relational operators evaluates to true or false based on collating sequence
- Example: `'R' > 'T'`

Expression	Value of the Expression	Explanation
`' ' < 'a'`	`true`	The Unicode value of `' '` is 32, and the Unicode value of `'a'` is 97. Because 32 < 97 is `true`, it follows that `' ' < 'a'` is `true`.
`'R' > 'T'`	`false`	The Unicode value of `'R'` is 82, and the Unicode value of `'T'` is 84. Because 82 > 84 is `false`, it follows that `'R' > 'T'` is `false`.
`'+' < '*'`	`false`	The Unicode value of `'+'` is 43, and the Unicode value of `'*'` is 42. Because 43 < 42 is `false`, it follows that `'+' < '*'` is `false`.
`'6' <= '>'`	`true`	The Unicode value of `'6'` is 54, and the Unicode value of `'>'` is 62. Because 54 <= 62 is `true`, it follows that `'6' <= '>'` is `true`.

Relational Operators

- Examples : Indicate T or F (1 or 0) for each of the following :

```
int x=6;
int y=10;

a. x == 5
b. 7 <= x + 2
c. y = 3 > x
d. x != y
e. true
```

- We Can assign relational exprs to variables:

```
bool isPositive, found;
int x;

cin >> x;
isPositive = x > 0;
found = x == 100;
```

- Relational ops have higher precedence than =

Precedence and Relational Operators

- Relational operators are **lower** than **arithmetic** operators :

```
int x, y;
... x < y -10 ... // minus happens first
... x * 5 >= y + 10 ... // mult, then plus, then >=
```

- Relational operators are **higher** than **assignment** :

```
int x, y;
...
bool t1 = x > 7; // > then =
bool t2 = x * 5 >= y + 10; // *, +, >=, =
```

if-else statement

- if-else statement is used to express decisions

```
if (expression)
statement1
else
statement2
```

- The expression is evaluated:
  - If it is true, then statement1 is executed.
    (statement2 is skipped).
  - If it is false, then statement2 is executed
    (statement1 is skipped).

if-else example

```
double rate;
double monthlySales;
cout << "Enter monthly sales last month: " ;
cin >> monthlySales;
if (monthlySales > 3000)
        rate = .025;
else
        rate = .029;
double price;
cout << "Enter selling price of item: " ;
cin >> price;
double commission = (price + 3.99) * rate;
cout  << "Commission:  $" << commission << endl;
```

Sample Output

```
Enter monthly sales last month: 5430
Enter selling price of item: 175
Commission:  $4.47475
```

Notice that :

```
if (monthlySales > 3000)
      rate = .025;
else
      rate = .029;
```

- Relational expression in parentheses
- NO semi-colon after expression, nor else
- Good style: indent the statements
- The semi-colons belong to the statements, not to the if-else

The block statement

- a block (or a compound statement) is a set of statements inside braces:

```
{
int x;
cout << "Enter a value for x: " << endl;
cin >> x;
cout << "Thank You  " << endl;
}
```

- This allows us to use multiple statements when by rule only one is allowed.

if-else with blocks

- We can use blocks to put more than one statement in the branches of the if-else:

```
int number;
cout << "Enter a Number : " << endl;
cin >> number;
if (number % 2 == 0)
{
        number = number / 2;
        cout << "Even" << endl;
}
else
{
        number = (number – 1) / 2;
        cout << "Odd " << endl;     }
```

if statement

- The else part is optional:

```
if (expression)
statement1
```

- Expression is evaluated:
  - If it is true, then statement1 is executed.
  - If it is false, then statement1 is skipped.

if statement example

- Example : Input Validation

```
cout << "Enter a positive number: ";
cin >> x;
if (x < 0)
{
        cout << "That number is negative. "
        << "Please enter a positive number: ";
        cin >> x;
}       //do something with x here
```

Watch out

- What is output of the following code segment ?

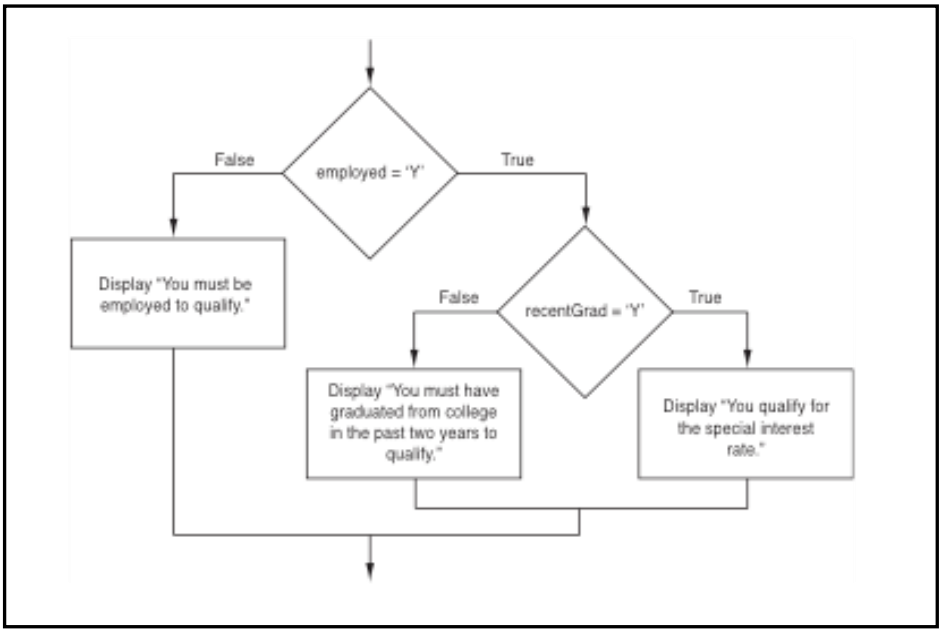
```
int x;
x = 13;
if (x==10)
      x = 17;
cout << x << endl;
cout << "Done !" << endl;
```

- What is output of the following code segment ?

```
char babyGender;
cin >> babyGender;
if (babyGender == 'M')
     cout << "It's a boy!" << endl;
cout << "It's a girl!" << endl;
```

Nested If statements

- if-else is a statement. It can occur as a branch of an if-else statement.



Example : Nested if statements

```
char bornInUSA;
int age;
cout << " Were you born in USA  ( Y / N )   " ;
cin >> bornInUSA;
cout << " What is your age; ";
cin >> age;
if (bornInUSA == 'Y')
        if (age >= 35)
                cout << "You qualify to run for President" << endl;
          else
             cout << "You are too young to run for President" << endl;
else
        cout << "You must have been born in the US in order " <<
        "to run for President" << endl;
```

Dangling Else Problem

- Combining an if with an if-else :

```
if (a > 0)
      if (b > 0)
          cout << "*****" << endl;
  else
          cout << "-----" << endl;
```

- Or is it:

```
if (a > 0)
      if (b > 0)
           cout << "*****" << endl;
else
           cout << "-----" << endl;
```

- It's the first one. The else is paired with the closest if.

To override the dangling else convention

- Add braces :

```
if (a > 0)
{
    if (b > 0)
       cout << "*****" << endl;
}
else
   cout << "-----" << endl;
```

Common nested if pattern

- Determine letter grade from testScore :

```
if (testScore < 60)
   grade = 'F';
else {
       if (testScore < 70)
           grade = 'D';
       else {
             if (testScore < 80)
                grade = 'C';
             else {
                   if (testScore < 90)
                       grade = 'B';
                   else
                       grade = 'A';
                  }
             }
      }
```

- Note that braces are actually optional here

if-else if (.. else-if)

- Not really a different statement, just a different way of indenting the previous nested if statement :

```
if (testScore < 60)
      grade = 'F';
else if (testScore < 70)
      grade = 'D';
else if (testScore < 80)
      grade = 'C';
else if (testScore < 90)
      grade = 'B';
else
      grade = 'A';
```

Comparing Floating-Point Numbers when using ==

Because of the nature of the floating points and the way they are stored in memory, rounding errors sometimes occur. That is because some of the fraction numbers can not be exactly represented using binary .

```
double a = 1.5 ;            //        a = 1.5
double b = 1.5 ;            //        b = 1.5
a += 0.0000000000000001;    // Adding a little bit value

if ( a == b )
        cout << "Both a  and  b  are equal ";
else
        cout << " a  and  b  are  not equal";
```

Try to stay with greater than or less than comparison with the floating-point numbers.

Logical (Boolean) Operators and Logical Expressions :

- Logical (Boolean) operators enable you to combine logical expressions
- Logical operators take logical values as operands
- Binary operators: && and || or
- Unary operator: !

Logical Operators :-

Logical Operator	Meaning	Evaluates
&&	AND	Both conditions must be true for the entire condition to be true
||	OR	Either condition or both conditions must be true for the entire condition to be true
!	NOT	Reverse the truth condition

Logical Operators Table (and &&)

Expression1	Expression2	Expression1 && Expression2
true	true	true
true	false	false
false	true	false
false	false	false

Logical Operators Table (or ||)

Expression1	Expression2	Expression1 || Expression2
true	true	true
true	false	true
false	true	true
false	false	false

Logical Operators :

Examples :

```
/*
 *      Author: Husain Gholoom
 *      Example : Logical Operators
 *
 */
#include<iostream>
using namespace std;
int main()
{
        bool found = true;
        bool flag = false;
        double x = 5.2, num1=2567.58,num2=2567.58;double y = 3.4;
        int a = 5;  int b = 8;  int n = 20;
        char ch = 'B';

         cout<< "!found evaluates to " << !found <<endl;
         cout<< "x > 4.0 evaluates to " << (x > 4.0) <<endl;
         cout<<"!found && (x >= 0) evaluates to "<<(!found && (x >= 0))<<endl;
         cout<<"!(found && (x >= 0)) evaluates to "<< !(found && (x >= 0))<<endl;
         cout<<"x + y <= 20.5 evaluates to " << (x + y <= 20.5)<<endl;
         cout<<"(n >= 0) && <<n <= 100) evaluates to " <<((n >= 0) && (n <= 100))<<endl;
         cout<<"('A' <= ch && ch <= 'Z') evaluates to " <<('A' <= ch && ch <= 'Z')<<endl;
         cout<<"(a + 2 <= b) && !flag evaluates to "<<((a + 2 <= b) && !flag)<<endl;
         cout<<"The value of num1 == num2 is " <<( num1 == num2)<<endl;
   return 0;
}
```

Precedence and Logical Operators

- ! is higher than most operators, so use parentheses:

```
int x;
... !(x < 0 && x > -10) ...
```

- && is higher than ||
-

```
int x, y;
...
... flag || x * 5 >= y + 10 && x == 5

// which op is first? second? etc?
```

- ! && and || are lower than arithmetic + relational Operators : parens **not** usually needed

Precedence	Symbols
Highest	* / %
	+ -
	> >= < <=
	= = !=
	&&
	| |
Lowest	=

Checking Numeric Ranges

- Want x to be in the range from 1 to 10 (inclusive)

```
a.    if (1 <= x <= 10)
      cout << "YES" << endl;
            //NO, which op is done first? second?


b.    if (1 <= x && x <= 10)
      cout << "YES" << endl;

      -     check: x=0?
      -     check: x=5?
      -     check: x=100?
```

Short Circuit Evaluation

- What is the value of : x != x && y > 10
  - true
  - false
  - don't have enough information to determine

- Actually it is false.
  - x != x is always false.
  - false && ?? is always false
    - false && false is false
    - false && true is false

- If expression on the left of && is false, the expression on the right is not evaluated, the result is false.

- If expression on the left of || is true, the expression on the right is not evaluated, the result is true.

Watch out

- What is output?

```
int x=10, y=15;
if (x+y)
   cout << "x+y is true. " << endl;
```

  - anything not 0 (zero) is true.
  - 0 (zero) is false.

- What is output?

```
int x;
cin >> x;
if (x = 5)
      cout << x << endl;
```

- It always outputs : 5. Why ??

- What is output?

```
double x = 1.0/9.0;
if (x+x+x + x+x+x + x+x+x == 1.0)
    cout << "Nine ninths is one" << endl;
else
   cout << "Uh Oh" << endl;
```

- Some fractional numbers cannot be stored exactly using binary (round-off errors)
- Computation can compound these errors.
- Don't test floating point values using equality

Compare using ***else-if*** to a series of ***if*** statements.

```
if ( month == 1 )
   cout << "Jan";
else if ( month == 2 )
   cout << "Feb";
else if ( month == 3 )
   cout << "Mar";
```

VS

```
if ( month == 1 )
   cout << "Jan";
if ( month == 2 )
   cout << "Feb";
if ( month == 3 )
   cout << "Mar";
```

Does Order Matter ??

```
if ( average >= 89.5 )
      ltr_grd = 'A' ;
else if ( average >= 79.5 )
      ltr_grd = 'B';
else if ( average >= 69.5 )
      ltr_grd = 'C';
else if ( average >= 59.5 )
      ltr_grd = 'D';
else
      ltr_grd = 'F';
```

VS

```
if ( average >= 69.5 )
      ltr_grd = 'C' ;
else if ( average >= 79.5 )
      ltr_grd = 'B';
else if ( average >= 89.5 )
      ltr_grd = 'A';
else if ( average >= 59.5 )
      ltr_grd = 'D';
else
      ltr_grd = 'F';
```

The Ternary Statement

There is another seldom-used selection construct in C++; the **ternary** statement. In its simplest form, it is very similar to the **if** statement in that a **true/false condition** determines which of two statements is executed. An example is shown below.

```
(x > y) ? cout << x << " is greater than " << y << endl:
          cout << y << " is greater than or equal to " << x << endl;
```

Note that the **condition** in parentheses is given first, which is followed by a **question mark** and the **statement to be executed** if the condition is **true**. This is separated from the **statement that will be executed** if the condition is **false** by a **colon**. The statement must, as always, be terminated by a **semicolon**.

1. What is the output of the following segment of code if the input value is 20?

```
cin >> someInt;
if (someInt > 30)
    cout << "Moe ";
    cout << "Larry ";
cout << "Curley ";
```

a. Curley b. Moe Larry Curley c. Larry Curley
d. There is no output, there is a compile-time error
e. There is no output, there is a run-time error

2. After execution of the following code, what will be the value of angle if the input value is 0?

```
cin >> angle;
if (angle>5)
    angle = angle + 5;
else if (angle > 2)
    angle = angle + 10;
else
    angle = angle + 15;
```

a. 15 b. 10 c. 25 d. 0 e. 5

3. What will be the output given the segment of code, if a 0 is entered from the keyboard?

```
int x = -1;
cout << " Enter a 0 or a 1 from the keyboard: ";
cin >> x;
if (x)
  cout << x;
```

a. the output will be a 0 b. nothing will be output
c. x will be outputted d. a –1 will be outputted
e. I do not know so cout it wrong!

4. After fixing all syntax errors, what is the output of thefollowing code segment if a 0 is entered from the keyboard?

```
int number;
cin << number;
if (number = 0);
  cout << "Moe" << endl;
else
  cout << "Larry" << endl
  cout << "Curley"<< endl;
```

a. Moe
Curley
b. Larry
Curley
c. Moe Curley
d. Larry Curley
e. Nothing, a compiler syntax error exists

5. What will be the output given the coding segment below?

```
int x=1, y=0;
 if (x == 1)
   cout << "Moe ";
 if (y = 1)
   cout << "Larry ";
 cout << "Curley ";
```

a. Moe Larry Curley
b. Larry Curley
c. Curley
d. Moe Curley
e. None of these

Program Problem 1: Assume that we Have 3 variables a, b, c. Write IF-Statements that find the variable that has the highest value and sets another variable, max to that value.

Program Problem 2 : Assume that we Have 3 variables a, b, c. Write an IF-Statement that sort them in an ascending order from highest to lowest.