Operator,

Operators are used to manipulate and check operand values.

Operand

Operands are numerical, text and Boolean values that a program can manipulate and use as logical guidelines for making decisions.  
  
Infix,

Operators are written in-between their operands. This is the usual way we write expressions. An expression such as A \* ( B + C ) / D is usually taken to mean something like: "First add B and C together, then multiply the result by A, then divide by D to give the final answer."

Infix notation needs extra information to make the order of evaluation of the operators clear: rules built into the language about operator precedence and associativity, and brackets ( ) to allow users to override these rules. For example, the usual rules for associativity say that we perform operations from left to right, so the multiplication by A is assumed to come before the division by D. Similarly, the usual rules for precedence say that we perform multiplication and division before we perform addition and subtraction.

postfix,

Operators are written after their operands.

The order of evaluation of operators is always left-to-right, and brackets cannot be used to change this order. Because the "+" is to the left of the "\*" in the example above, the addition must be performed before the multiplication.   
Operators act on values immediately to the left of them. For example, the "+" above uses the "B" and "C". We can add (totally unnecessary) brackets to make this explicit:   
( (A (B C +) \*) D /)   
Thus, the "\*" uses the two values immediately preceding: "A", and the result of the addition. Similarly, the "/" uses the result of the multiplication and the "D".

prefix notation

Operators are written before their operands. The expressions given above are equivalent to / \* A + B C D   
As for Postfix, operators are evaluated left-to-right and brackets are superfluous. Operators act on the two nearest values on the right. I have again added (totally unnecessary) brackets to make this clear:   
(/ (\* A (+ B C) ) D)

Precedence and associativity

Precedence rules specify that certain operators, in the absence of parentheses,

group “more tightly” than other operators. Associativity rules specify that sequences of operators of equal precedence group to the right or to the left.

How are precedence and associativity implemented in the parse tree.

Assignment as an expression

 an **assignment statement** sets and/or re-sets the [value](http://en.wikipedia.org/wiki/Value_(computer_science)) stored in the storage location(s) denoted by a [variable](http://en.wikipedia.org/wiki/Variable_(programming)) [name](http://en.wikipedia.org/wiki/Name); in other words, it copies the value into the variable. In most [imperative](http://en.wikipedia.org/wiki/Imperative_programming) [programming languages](http://en.wikipedia.org/wiki/Programming_language), the assignment statement (or expression) is a fundamental construct.

Value model, reference model, functions that return references, aliases

Value model

The memory address contains the value of the data

Reference model

The memory address contains the address of the data

Functions that return references

aliases

Boxing,

Boxing, otherwise known as wrapping, is the process of placing a primitive type within an object so that the primitive can be used as a reference object.

wrapper classes,

These are known as wrapper classes, because they "wrap" the primitive data type into an object of that class.

implicit versus explicit boxing

which means you don't need to tell the compiler that you are boxing Int32 to object because it takes care of this itself although you can always make explicit boxing as seen below right after implicit boxing.

Int32 x = 5;

object o = x; *// Implicit Boxing*

x = (Int32)o; *// Explicit UnBoxing*

Operand order of evaluation, Java model

http://introcs.cs.princeton.edu/java/11precedence/

Short circuit operators, advantages and disadvantages

Untested second condition leads to unperformed side effect

Avoiding the execution of second expression's side effects

What are the 8 control flow mechanisms?

http://en.wikipedia.org/wiki/Flow\_control\_%28data%29

Assignment, assignment as an expression, dangers in C with decisions, dangers in C++ decisions with implicit promotion

Assignment operators =, auto-increment, auto-decrement

Initialization,

dynamic checking,

Instead of giving every uninitialized variable a default value, a language or implementation

can choose to define the use of an uninitialized variable as a dynamic

semantic error, and can catch these errors at run time.

definite assignment,

Java and C# require that a value be “definitely assigned” to a variable before that

variable is used in any expression. Both languages provide a precise definition of

“definitely assigned,” based on the control **EXAMPLE** 6.26 flow of the program.

Constructors

In [class-based](http://en.wikipedia.org/wiki/Class-based_programming) [object-oriented programming](http://en.wikipedia.org/wiki/Object-oriented_programming), a **constructor** (abbreviation: **ctor**) in a [class](http://en.wikipedia.org/wiki/Class_(computer_programming)) is a special type of [subroutine](http://en.wikipedia.org/wiki/Subroutine" \o "Subroutine)called to [create an object](http://en.wikipedia.org/wiki/Object_creation). It prepares the new object for use, often accepting [arguments](http://en.wikipedia.org/wiki/Argument) that the constructor uses to set required [member variables](http://en.wikipedia.org/wiki/Member_variable).

Multiway assignment

which simultaneously assigns 0 to a and 1 to b. This is most often known as **parallel assignment**;