**Expressions** – are the fundamental means of specifying computations in a PL.

**Expression evaluation** – orders of operator and operand evaluation is needed which is associativity and precedence.

**Issues in expression semantics**

1. type mismatches
2. coercions
3. short-circuit evaluation

**Assignment statements** – purpose to change the value of a variable.

**Arithmetic expressions** – consist of operators, operands, parentheses, and function calls. Its purpose is to specify an airthmetic computation.

**Operators** - can be unary, binary, and ternary. Binary operators used by PLs are infix except for Perl which uses prefix.

**Precedence** – concept of placing operators in a heirarchy of evaluation priorities.

**Associativity** – an operator can either have left or right associativity.

**Parentheses** – way to alter the precedence and associaitivity rules

**Side effects** – called functional side effects, occur when the function changes either one of its parameters or a global variable.

**1st solution:** disallow function evaluation from affecting the value of expressions by disallowing functional side effect

**2nd solution:** state the language definition that operands in expressions are to be evaluated in a particular order.

**Operator overloading** – is using an arithmetic operator for more than one purpose.

**Type conversion**

1. Narrowing – converts that cannot store even approximation (double to float)
2. Widening – converts that include at least approximation (int to float)

**Coercions** – implicit operand type conversion

**Type Casting** – explicit type conversion

**Wide range of coercion -** affects reliability since type checking is eliminated.

**Smaller range of coercion** - affects flexibility since a lot of restriction exist.

**Errors in expressions:**

1. Erros can occur in expressions due to coercions
2. Limitiations of computer arithmetic (overflow and underflow)
3. Inherent limitations of arithmetic (division by zero)

**Relational operator** – compares the values of its two operands. Have lower precedence than the arithmetic operator.

**Relational expression** – has two operands and one relational operator

**Typical operand types:** numeric, strings, ordinal types

**Short-circuit evaluation** – is one in which the result is determined without evaluating all the operands / operators.

**Control statements** – means of selecting among alternative control flow paths

**Control structure** – is a control statement and the collection of statements whose execution it controls

**Selection statement** – provides the means of choosing between two or more execution paths in a program

**Overloaded subprogram**

* has the same name as another subprogram
* Must have a unique protocol (number, order, types of parameter, return type)

**C++, Java and Ada**

* Include predefined overloaded subprograms
* Same subprogram names but different parameters

**Generic subprograms**

* Software reusecan be an important contributor to software productivity increase
* Lessens the need to create different subprograms that implement same algorithm and different types of data
* Takes parameters of different types on different activations

**Parametric polymorphism**

* Is provided by a subprogram that takes a generic parameter that is used in a type expressions that describes the types of the parameters of the subprogram

**Separate and Independent Compilation**

* The capability of compiling parts of a program without compiling the whole program is essential to the construction of large software systems.

**Linker**

* Collects the newly compiled and previously compiled units

**Separate Compilation**

* Means that the compilation units can be compiled at different times but not independent.
* Must have access to information about program entities (variable, types, protocol of subprograms)

**Independent Compilation**

* Program units can be compiled without information about any other subprogram units
* Interfaces between the separately compiled units are not checked for consistency

**Separate and Independent**

* Neither separate nor independent
* Fortran II and Pascal

**Design Issues for functions**

* Are the side effects allowed?
* What types of values can be returned?

**Accessing Nonlocal Environment**

* Can be accomplished through parameters
* provide through accessing variables from external environment

**Nonlocal variable**

* Are those visible within the subprogram but not locally declared

**Global variables**

* Are those visible in all program units

**Problems**

* Local variables of the subprogram are visible to any other executing subprograms regardless of its textual proximity
* Inability to statically type-check references of nonlocals.

**Coroutines**

* Is a special kind of subprogram
* Rather than master-slave relationship between caller and called subprogram, the two are more equal basis
* Coroutine control mechanism is often called symmetric control unit model.