Chapter 4

Introduction

Expressions are the fundamental means of specifying computations in a programming language.

Arithmetic Expressions

Arithmetic evaluation was one of the motivations for the development of the first programming language.

Arithmetic Expressions Design issues

- Operator precedence rules?
- Operator associativity rules?
- Order of operand evaluation?
- Operand evaluation side effects?
- Operator overloading?
- Type mixing in expressions?

Operators

Unary operator has one response

Binary operator has two response

Ternary operator has three response

Operator Precedence Rules

- Parentheses.
- Unary operators
- ** (if language supported)
- * , /
- + , -

Conditional Expressions

- C-based languages (like C, C++)

An example:

```
average = (count == 0)? 0 : sum / count
```

Evaluates as if written like

```
if (count == 0)
   average = 0
else
   average = sum /count
```

Operand Evaluation Order

- **Variables**: fetch value from memory
- **Constants**: maybe fetched from memory or it's machine language instruction.
- **Parenthesized expressions**: evaluate all operands and operators first.

Chapter 4

- An interesting case is when operand is a function call

Overloaded Operators

It's Use of an operator for more than one purpose

Some of them are common such as **Integer,**

Others have problems such as ***** in **C** and **C++**

Type conversation

Narrowing conversation: convert an object to a type that cannot include all of the value of the original Type.

Ex: float to int

Widening conversion: object is converted to a type that at least approximation to all of the value of the original type.

Ex: int to float

Mixed Mode

it has operand of different Types

coercion: implicit type conversion.

Errors in Expressions

- Inherent limitations of arithmetic.
  Such as division by zero
- Limitations of computer arithmetic.
  Sush as overflow

Relational and Boolean Expressions

Relational Expressions:-

- Use relational operators and operand of various types.
- Evaluate to some Boolean representation.
- Operator symbol used vary somewhat among languages (!=,/=,~=,.NE.,<>,#)

FORTAN77	FORTAN90	C	Ada
.AND.	and	&&	and
.OR.	or	||	or
.NOT.	not	!	not
			xor

Chapter 4

Short Circuit Evaluation

an expression in which the result is determined without evaluating all of the operands and/or operators

such as: (13*a) * (b/13–1)

C, C++, and Java: use short-circuit evaluation for the usual Boolean operators (&& and ||), but also provide bitwise Boolean operators that are not short circuit (& and |).

Assignment Statements

Conditional Targets

Conditional targets (Perl)

```
($flag ? $total : $subtotal) = 0
```

Which is equivalent to

```
if ($flag){
  $total = 0
} else {
  $subtotal = 0
}
```

Compound Operators

- A shorthand method of specifying a commonly needed form of assignment
- Introduced in ALGOL; adopted by C
- Example

```
a = a + b
```

is written as

```
a += b
```

Assignment as an Expression

In C, C++, and Java, the assignment statement produces a result and can be used as operands