Operators & Expressions

EECS 2031

Song Wang
wangsong@eecs.yorku.ca
eecs.yorku.ca/~wangsong/

Acknowledgement

- Some of the covered materials are based on previous EECS2031 offerings:
  - Uyen Trang (UT) Nguyen, Pooja Vashisth, Hui Wang, Manos Papagelis

Definition

“An operator is a symbol (+,-,*,/) that directs the computer to perform certain mathematical or logical manipulations and is usually used to manipulate data and variables”

Ex: a+b

Operators in C

1. Arithmetic operators
2. Relational operators
3. Logical operators
4. Assignment operators
5. Increment and decrement operators
6. Conditional operators
7. Bitwise operators
8. Special operators

Arithmetic operators

Operator	example	Meaning
+	a + b	Addition –unary
-	a – b	Subtraction- unary
*	a * b	Multiplication
/	a / b	Division
%	a % b	Modulo division-remainder

Relational Operators

Operator	Meaning
<	Is less than
<=	Is less than or equal to
>	Is greater than
>=	Is greater than or equal to
==	Equal to
!=	Not equal to

Logical Operators

Operator	Meaning
&&	Logical AND
||	Logical OR
!	Logical NOT

Logical expression or a compound relational expression

An expression that combines two or more relational expressions

Ex: if (a==b && b==c)

Truth Table

Shows **truth value** of a compound statement for all possible truth values of the component statements

If there are *n* component statements, then the truth table has **2^n rows**

a	b	Value of the expression	
		a && b	a || b
F	F	F	F
F	T	F	T
T	F	F	T
T	T	T	T

Connectives and Symbols

Connective	Symbol	Type of Statement
and	&&, &	conjunction
or	||, |	disjunction
not	!, ~	negation

Conjunction

p	q	p & q
T	T	T
T	F	F
F	T	F
F	F	F

Disjunction

p	*q*	*p* | *q*
T	T	T
T	F	T
F	T	T
F	F	F

Negation

p	$\sim p$
T	F
F	T

Assignment operators

Syntax:

v op = exp;

Where v = variable,

op = shorthand assignment operator

exp = expression

Ex: x=x+3

x+=3

Operator	Description	Example
=	Simple assignment operator. Assigns values from right side operands to left side operand	C = A + B will assign the value of A + B to C
+=	Add AND assignment operator. It adds the right operand to the left operand and assign the result to the left operand.	C += A is equivalent to C = C + A
-=	Subtract AND assignment operator. It subtracts the right operand from the left operand and assigns the result to the left operand.	C -= A is equivalent to C = C - A
*=	Multiply AND assignment operator. It multiplies the right operand with the left operand and assigns the result to the left operand.	C *= A is equivalent to C = C * A
/=	Divide AND assignment operator. It divides the left operand with the right operand and assigns the result to the left operand.	C /= A is equivalent to C = C / A
%=	Modulus AND assignment operator. It takes modulus using two operands and assigns the result to the left operand.	C %= A is equivalent to C = C % A
<<=	Left shift AND assignment operator.	C <<= 2 is same as C = C << 2
>>=	Right shift AND assignment operator.	C >>= 2 is same as C = C >> 2
&=	Bitwise AND assignment operator.	C &= 2 is same as C = C & 2
^=	Bitwise exclusive OR and assignment operator.	C ^= 2 is same as C = C ^ 2
|=	Bitwise inclusive OR and assignment operator.	C |= 2 is same as C = C | 2

Shorthand Assignment operators

Simple assignment operator	Shorthand operator
a = a+1	a + =1
a = a-1	a - =1
a = a* (m+n)	a * = m+n
a = a / (m+n)	a / = m+n
a = a %b	a %=b

Increment & Decrement Operators

C supports 2 useful operators namely

1. Increment ++

2. Decrement - - operators

The ++ operator adds a value 1 to the operand
The - - operator subtracts 1 from the operand
++a or a++
--a or a- -

Rules for ++ & -- operators

1. These require variables as their operands

2. When postfix either ++ or - - is used with the variable in a given expression, the expression is evaluated first and then it is incremented or decremented by one

3. When prefix either ++ or - - is used with the variable in a given expression, it is incremented or decremented by one first and then the expression is evaluated with the new value

Examples for ++ & -- operators

Let the value of a =5 and b=++a then

a = b =6

Let the value of a = 5 and b=a++ then

a =6 **but** b=5

i.e.:

1. a prefix operator first adds 1 to the operand and then the result is assigned to the variable on the left
2. a postfix operator first assigns the value to the variable on left and then increments the operand.

Conditional operators

Syntax:

exp1 ? exp2 : exp3

Where exp1,exp2 and exp3 are expressions

Working of the ? Operator:

Exp1 is evaluated first, if it is nonzero(1/true) then the expression2 is evaluated and this becomes the value of the expression,

If exp1 is false(0/zero) exp3 is evaluated and its value becomes the value of the expression

Ex: m=2;

```
n=3
r=(m>n) ? m : n;
```

Bitwise operators

These operators allow manipulation of data at the bit level

Operator	Description
&	Binary AND Operator copies a bit to the result if it exists in both operands.
|	Binary OR Operator copies a bit if it exists in either operand.
^	Binary XOR Operator copies the bit if it is set in one operand but not both.
~	Binary One's Complement Operator is unary and has the effect of 'flipping' bits.
<<	Binary Left Shift Operator. The left operands value is moved left by the number of bits specified by the right operand.
>>	Binary Right Shift Operator. The left operands value is moved right by the number of bits specified by the right operand.

eXclusive OR

- XOR compares two input bits and generates one output bit. If the bits are the same, the result is 0. If the bits are different, the result is 1

x_1	x_2	x_1 XOR x_2
0	0	0
0	1	1
1	0	1
1	1	0

```c
#include <stdio.h>

main() {

   unsigned int a = 60; /* 60 = 0011 1100 */
   unsigned int b = 13; /* 13 = 0000 1101 */
   int c = 0;

   c = a & b;       /* 12 = 0000 1100 */
   printf("Line 1 - Value of c is %d\n", c );

   c = a | b;       /* 61 = 0011 1101 */
   printf("Line 2 - Value of c is %d\n", c );

   c = a ^ b;       /* 49 = 0011 0001 */
   printf("Line 3 - Value of c is %d\n", c );

   c = ~a;          /*-61 = 1100 0011 */
   printf("Line 4 - Value of c is %d\n", c );

   c = a << 2;     /* 240 = 1111 0000 */
   printf("Line 5 - Value of c is %d\n", c );

   c = a >> 2;     /* 15 = 0000 1111 */
   printf("Line 6 - Value of c is %d\n", c );
}
```

Misc Operators

Operator	Description	Example
sizeof()	Returns the size of a variable.	sizeof(a), where a is integer, will return 4.
&	Returns the address of a variable.	&a; returns the actual address of the variable.
*	Pointer to a variable.	*a;
? :	Conditional Expression.	If Condition is true ? then value X : otherwise value Y

Rules for evaluation of expression

1. First parenthesized sub expressions from left to right are evaluated.

2. If parentheses are nested, the evaluation begins with the innermost sub-expression

3. The precedence rule is applied in determining the order of application of operators in evaluating sub-expressions

4. The associative rule is applied when 2 or more operators of the same precedence level appear in a sub expression.

5. Arithmetic expressions are evaluated from left to right using the rules of precedence

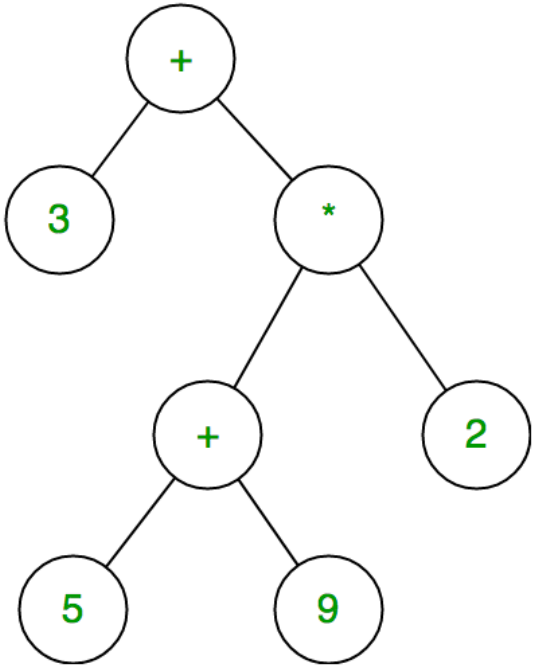
6. When parentheses are used, the expressions within parentheses assume highest priority

Order of operators

Operator	Description	Associativity
( ), [ ]	Function call, array element reference	Left to Right
+, -, ++, --,!,~,*,&	Unary plus, minus, increment, decrement, logical negation, 1's complement, pointer reference, address	Right to Left
*, / , %	Multiplication, division, modulus	Left to Right

Expression Tree

- A binary tree in which each internal node corresponds to the operator and each leaf node corresponds to the operand
  - 3 + ((5+9)*2)



Example

Evaluate x1=(-b+ sqrt (b*b-4*a*c))/(2*a) @ a=1, b=-5, c=6

=(-(-5)+sqrt((-5)*(-5)-4*1*6))/(2*1)

=(5 + sqrt((-5)*(-5)-4*1*6))/(2*1)

=(5 + sqrt(25 -4*1*6))/(2*1)

=(5 + sqrt(25 -4*6))/(2*1)

=(5 + sqrt(25 -24))/(2*1)

=(5 + sqrt(1))/(2*1)

=(5 + 1.0)/(2*1)

=(6.0)/(2*1)

=6.0/2 = 3.0