Operators and Expressions:

Operators and expression using numeric and relational operators, mixed operands, type conversion, logical operators, bit operations, assignment operator, operator precedence and associativity.

Objectives

- ♥ To be able to construct and evaluate expressions.
- ♥ To master operator precedence and associativity
- ♥ To understand implicit type conversion and explicit type conversion.

Introduction

- An operator is a symbol that tells the computer to perform certain manipulations.
- An expression is a sequence of operands and operators that reduces to a single value.
- C operators can be classified into a number of categories.
  - Arithmetic operators
  - Relational operators
  - Logical operators
  - Assignment operators
  - Increment and decrement operators
  - Conditional operators
  - Bitwise operators
  - Special operators

Arithmetic operators

- The arithmetic operators in C

Operator	meaning
+	Addition or unary plus
-	Subtraction or unary minus
*	Multiplication
/	Division
%	modulo division

Arithmetic operators

- Note:,
  - Integer division truncates remainder
  - The % operator cannot be applied to a float or double.
  - The precedence of arithmetic operators
    - Unary + or -
    - * / %
    - + -

```
#include<stdio.h>
main()
{
   int  months , days ;
   printf ( "Enter days \n “ ) ;
   scanf ( "%d“ , &days ) ;
   months = days / 30 ;
   days = days % 30 ;
   printf ( "Months =%d Days= %d \n", months, days);
}
```

Arithmetic expressions

- An arithmetic expression is a combination of variables, constants, and operators.
- For example,
- a*b-c → a*b-c
- (m+n)(x+y) → (m+n)*(x+y)
- ax^2+bx+c → a*x*x+b*x+c

Mathematical functions

- Mathematical functions such as cos, sqrt, log, etc. are frequently used in analysis of real-life problems. Table 3.9 on page 72 lists some standard math functions.

- We should include the line

#include<math.h>

in the beginning of the program.

Mathematical functions

- for example
- the roots of $ax^2+bx+c=0$ are

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

- x1= (-b + sqrt (b*b - 4*a*c)) / (2 * a)
- x2= (-b – sqrt (b*b - 4*a*c)) / (2 * a)

Relational Operators

- The relational operators in C are :

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

Relational Operators

- A relational expression yields a value of 1 or 0.
  - 5 < 6 1
  - -34 + 8 > 23 - 5 0
  - if a=3, b=2, c =1; then a > b > c is ?
- the associativity of relational operators is left → right
- table3.8 on page71

Relational Operators

- The relational operators >, >=, < and <= have the same precedence. Just below them in precedence are the equality operators: = = and !=.
- Relational operators have lower precedence than arithmetic operators , so an expression like i < lim-1 is taken as i < (lim-1).

- 3 >= 2 = = -4 < 0

Logical operators

- C has the following three logical operators
  - && meaning logical and
  - || meaning logical or
  - ! meaning logical not (unary operator)
- Expressions connected by && or || are evaluated left to right, and evaluation stops as soon as the truth or falsehood of the result is known.

Logical operators

op-1	op2	op-1&&op-2	op-1‖op-2	!op-1
Non-zero	Non-zero	1	1	0
Non-zero	0	0	1	0
0	Non-zero	0	1	1
0	0	0	0	1

Logical operators

- The precedence of && is higher than that of ||, and both are lower than relational operators, so

- 3 < 5 && -3 < -5 || 3 > 2
- char ch; to decide whether ch is an uppercase,

  ch >= ‘A’ && ch <= ‘Z’
- to decide whether a year is leap year,

  year% 4 ==0 && year % 100 != 0 || year % 400 = = 0

Assignment operators

- Assignment operators are used to assign the result of an expression to a variable.
- C has a set of 'shorthand' assignment operators of the form

  v op= exp;

- where v is a variable, exp is an expression and op is a C binary arithmetic operator. The operator op= is known as the shorthand assignment operator.
- The assignment statement v op= exp; is equivalent to v = v op (exp) ;
- For example, x += y + 1; This is same as the statement x = x + (y + 1) ;

Assignment operators

Statement with simple assignment operator	Statement with shorthand operator
a = a+1	a += 1
a = a-1	a -= 1
a = a * (n+1)	a *= n+1
a = a / (n+1)	a /= n+1
a = a % b	a %= b

Assignment operators

- The use of shorthand assignment operators has three advantages:
  - 1. What appears on the left-hand side need not be repeated and therefore it becomes easier to write.
  - 2. The statement is more concise and easier to read.
  - 3. The statement is more efficient.

Assignment operators

- int a=12, n=5;
  - (1) a += a
  - (2) a -= 2
  - (3) a *= 2 + 3
  - (4) a /= a + a
  - (5) a % = (n %= 2)
  - (6) a += a -= a *= a

Increment and decrement operators

- C provides two unusual operators for incrementing and decrementing variables.
- The increment operator ++ adds 1 to its operand, while the decrement operator -- subtracts 1.
- The unusual aspect is that ++ and -- may be used either as prefix operators (before the variable, as in ++n), or postfix operators (after the variable: n++).
- In both cases, the effect is to increment n. But the expression ++n increments n *before* its value is used, while n++ increments n *after* its value has been used.

- for example, if n is 5, then

```
x = n++;
```

is equivalent

```
x = n; n++;
```

but

```
x = ++n;
```

is equivalent

```
n++; x = n;
```

- The increment and decrement operators can only be applied to variables; an expression like (i+j)++ is illegal.

- The increment and decrement operators can be used in complex statements. Example:

  m=n++ -j +10;

- Consider the expression

  m = - n++ ;

- The precedence of ++ and – operators are the same as those of unary + and -.
- The associatively of them is right to left.
- m = - n++; is equivalent to m = - (n++)

- suppose,
- int a, b, c ; a = b = c = 1;
- After execution the following statements, what are the values of the expression and variables.
- (1) a>b && b>c++;
  (2) a-- || c++;
  (3) !a && b++;
  (4) ++a && ++b && ++c;
  (5) ++a && --b && ++c;

Conditional operator

- a ternary operator pair “? : ” is available in C to construct conditional expressions of the form

  expr1 ? expr2 : expr3

- the expression *expr1* is evaluated first. If it is non-zero (true), then the expression *expr2* is evaluated, and that is the value of the conditional expression. Otherwise *expr3* is evaluated, and that is the value. Only one of *expr2* and *expr3* is evaluated.

Special operators

- **1. The Comma Operator**
- The comma operator can be used to link the related expressions together. A comma-linked list of expressions is evaluated left to right and the value of right-most expression is the value of the combined expression. For example, the statement
- value = (x=10, y=5, x+y);
- first assigns the value 10 to x, then assigns 5 to y, and finally assigns 15 to value. Since comma operator has the lowest precedence of all operators, the parentheses are necessary.

- **2. The sizeof Operator**
- The sizeof is a compile time operator and, when used with an operand, it returns the number of bytes the operand occupies. The operand may be variable, a constant or a data type qualifier. Examples:
- m = sizeof (sum);
- n = sizeof (long int);
- k = sizeof (235L);

Example 3.3

The different kinds of operators.

```
#include<stdio.h>
main()
{
        int  a, b, c, d;
        a = 15;
        b = 10;
        c = ++a - b;
        printf ("a=%d b=%d c=%d\n", a, b, c );
        d = b++ +a;
        printf("a = %d b = %d d = %d\n", a, b, d );
        printf("a/b = %d\n", a / b);
        printf("a%%b = %d\n", a%b );
        printf("a *= b = %d\n", a *= b );
        printf("%d\n", ( c>d ) ? 1 : 0 );
        printf("%d\n", ( c<d ) ? 1 : 0 );
}
```

example3.4 variables in expressions and their evaluation

```
#include<stdio.h>
main()
{
   float a, b, c, x, y, z;
   a = 9;
   b = 12;
   c = 3;
   x = a – b / 3 + c * 2 - 1;
   y = a - b / ( 3+ c ) * ( 2 - 1 );
   z = a - ( b / ( 3 + c ) * 2 ) - 1;
   printf ( "x = %f \n", x );
   printf(" y = %f \n", y ) ;
   printf( "z = %f \n", z ) ;
}
```

Some Computational Problems

- When expressions include real values, then it is important to take necessary precautions to guard against certain computational errors. For example, consider the following statements:
  - a = 1.0 / 3.0;
  - b = a * 3.0;
- There is no guarantee that the value of b will equal 1.
- Another problem is division by zero.
- The third problem is to avoid overflow and underflow errors.

example3.5 round-off errors in computation of floating point numbers.

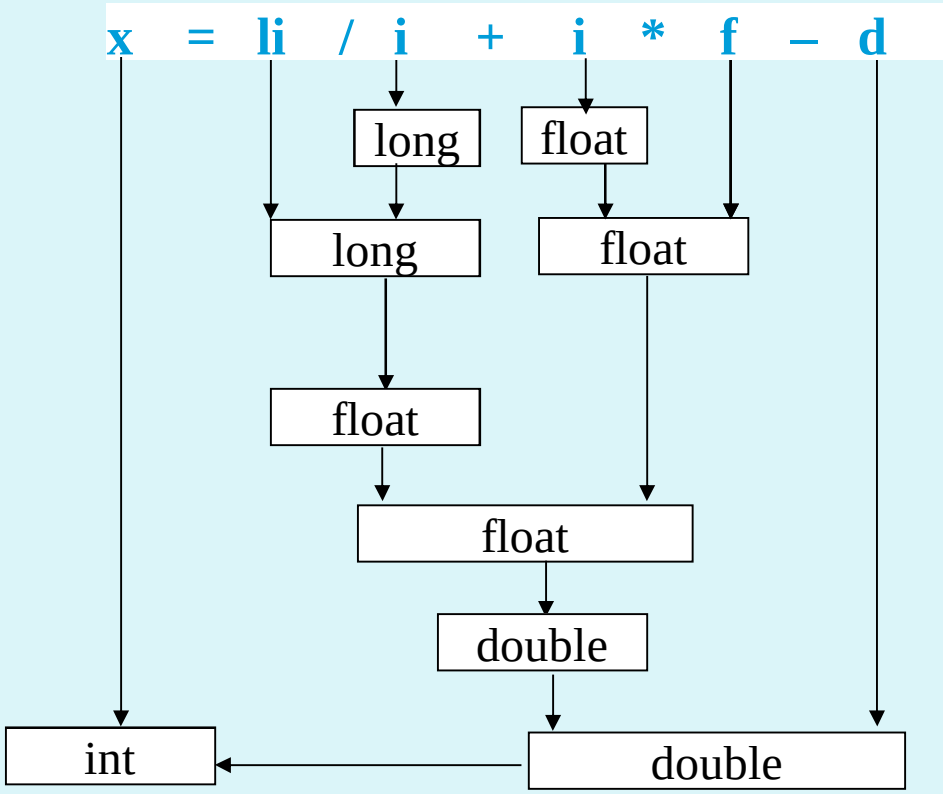
```
#include <stdio.h>
main()
{
    float sum, n, term;
    int count = 1;
    sum = 0;
    printf ( "Enter value of n \n " ) ;
    scanf ( "%f", &n ) ;
    term = 1.0 / n;
    while ( count <= n )
    {
            sum = sum + term ;
            count++ ;
    }
    printf ( "Sum = %f \n", sum ) ;
}
```

Type conversions in expressions

- **1. Implicit Type Conversion**
- C permits mixing of constants and variables of different types in an expression. C automatically converts any intermediate values to the proper type so that the expression can be evaluated without loosing any significance. This automatic conversion is known as implicit type conversion.
- The rule of type conversion: the lower type is automatically converted to the higher type.

Type conversions in expressions

- for example,
  - int i, x;
  - float f;
  - double d;
  - long int li ;

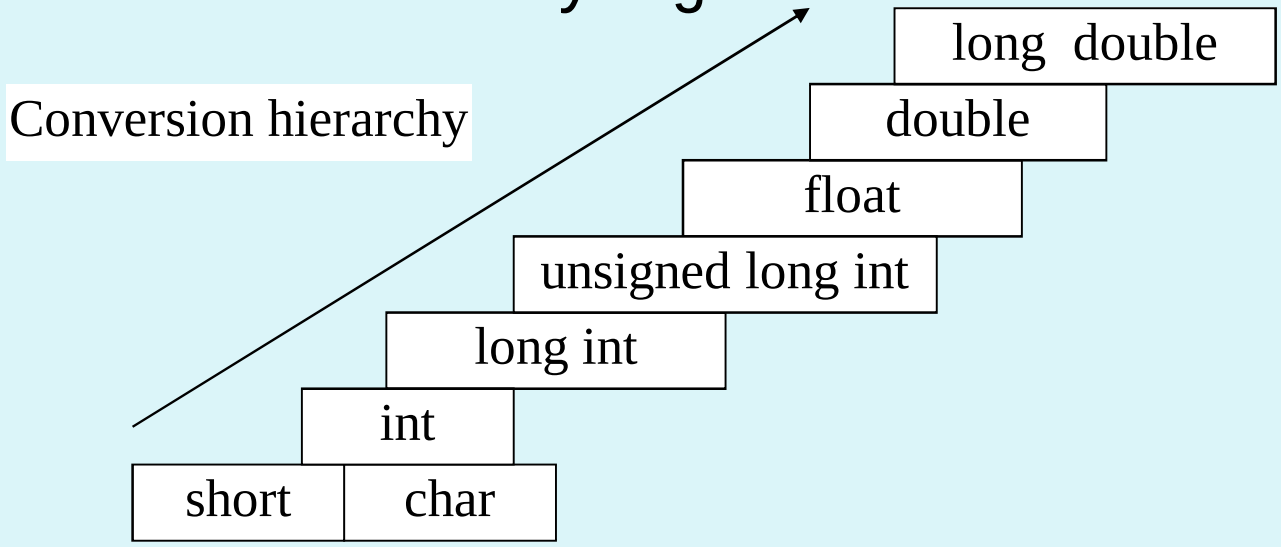


- The final result of an expression is converted to the type of the variable on the left of the assignment.

Type conversions in expressions

- The sequence of rules is given on page 67.

- Conversion Hierarchy is given below



Type conversions in expressions

- **2. Explicit conversion**
- We can force a type conversion in a way .
- The general form of explicit conversion is

  (type-name) expression

- for example
  - x = (int) 7.5 ;
  - a = (int) 21.3 / (int) 4.5;
  - a = (float) 3 / 2 ;
  - a = float (3 / 2) ;

Operator precedence and Associativity

- Rules of Precedence and Associativity
  - (1)Precedence rules decides the order in which different operators are applied.
  - (2)Associativity rule decide the order in which multiple occurrences of the same level operator are applied.
- Table3.8 on page71 shows the summary of C Operators.
- for example,
- **a = i +1== j || k and 3 != x**

Mathematical functions

- Mathematical functions such as cos, sqrt, log, etc. are frequently used in analysis of real-life problems.
- Table 3.9 on page 72 lists some standard math functions.
- All mathematical functions implement double type parameters and return double type values.
- To use any of mathematical functions, we should include the line:

#include < math. h >