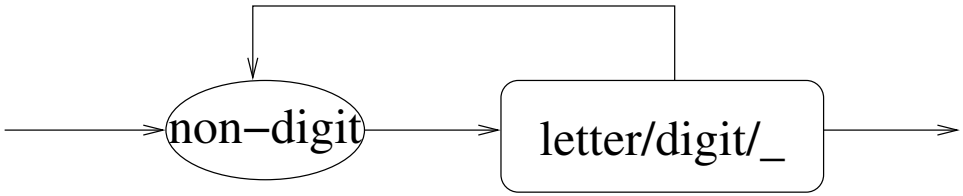
Expressions

Variables



The above rules indicates that the identifier must begin either a letter or underscore (non-digit) followed by 0 or more digits/letters.

Expressions

Possible identifiers

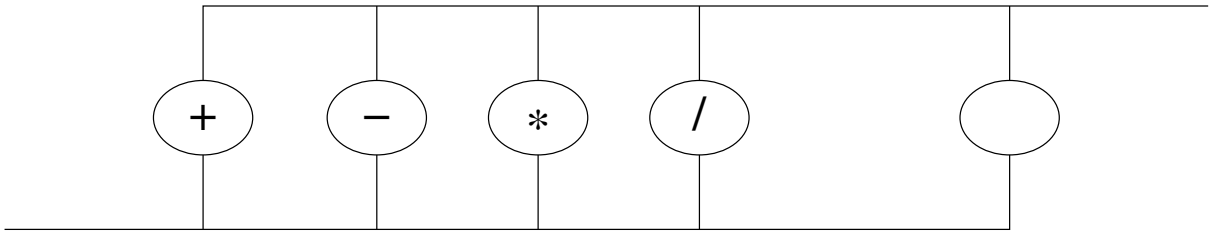
Valid	Non valid
`this This`	`1this`
`this15 T1his5`	`th$is15`
`this_is_car, thatIsAhouse`	`T@1his5`
`_This_is_15`	`_%this`
`_15_is_a_Number`	

- `printf` and `scanf` can be used for defining identifiers.
- But no reserved words.

Expressions

Operators

- **Unary operators**: operate on one operand, eg., &, ++, –, - (dual)
- **Binary operators**: most operators belong to this class.



- **Comma operator**: evaluates each expression separated by comma but returns the rightmost.
- **Conditional operator**: requires 3 operands
  `< condition > ? statement 1 : statement 2`

Expressions

Comma, and Conditional Operators

```
int total;
int i = 5, j = 10; k = 15;

total = (i + j, j + k, i + k);
printf("Total = %d\n", total);
```

`total` will return the value 20 (`i + k`)

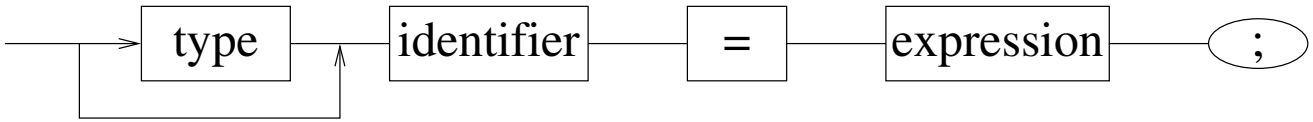
```
int z, a, b;

z = (a > b) ? a : b;
```

Sets `z` to `max(a, b)`

lvalue and Assignment Operator

lvalue



- Requires an **lvalue** as its left operand.
- L-value: represents an object stored in memory, which is neither a constant nor a result of computation.
- So a variable can be an **lvalue**, but neither any expressions or or any constant.

```
12 = i;          // WRONG
i + j = 0;       // WRONG
-i = j;          // WRONG
i++ = j;         // WRONG
```

lvalue and Assignment Operator

Compound Assignments

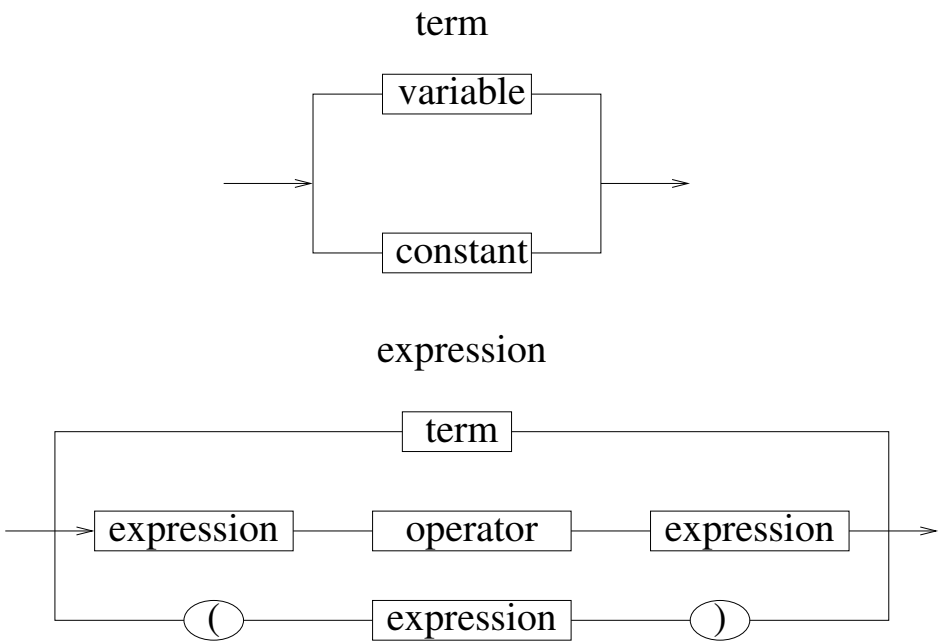
- Allows combination arithmetic operator with assignment. `+=`, `-=`, `*=`, `/=`, `%=`
- Multiple assignment also possible: `i = j = 10`
- Similarlly multiple compound assignments are possible where. evaluation is right associative, ie.,
  `i += j += k;` means `i += (j += k);`

Increment & Decrement

- Increment adds 1 to operand. Decrement substracts 1 from operand.
- Postfix and prefix of operator is possible.
- Postfix increments after use of the value
- Prefix increments before use of the value
- Left associative: eg. `a = i++ + j++;` if `i = 1, j = 2`, then `a`$\longrightarrow$ `3`
- Whereas, in `a = ++i + ++j;` `a = 0` means `i`$\longrightarrow 0$, `j`$\longrightarrow 0$. if `i, j` are non-negative.

Precedence order

Syntax Diagram of Expression



Precedence order

Precedence	Operator	Symbol	Associativity
1	Increment (postfix)	++	left
	Decrement (postfix)	–	left
2	Increment (prefix)	++	right
	Decrement (prefix)	–	right
	Unary plus	+	right
3	Unary minus	-	right
4	multiplicative	*, /, %	left
5	assignment	=, +=, etc.	right

Precedence order

Using expressions

Consider the code

```
a = b += c++ - d + --e/-f;
```

Highest precedence is for `c++`

Next in precedence order are: `--e` and `-f`

So putting parentheses in that order around the expressions:

```
a = b += (c++) - d + (--e)/(-f);
```

And finally, full parenthetic expression will be

```
(a = (b += (((c++) - d) + ((--e)/(-f)))));
```

With `a=1, b=2, c=12, d=2, e=5, f=2`, it evaluates: $a = 10$, $b = 10$

Precedence order

Using experssions (contd)

- `-a + (c + b * (c + a) / c - b / a) + a - b / 2` will be evaluated as
  `(((-a) + ((c + ((b * (c + a)) / c)) - (b / a))) - (b/2))`
- Assume `int i = 5, j = 10, k = 2, result;`
  - Then value `result = 2 * i % 5 * 4 + (j - 3) / (k + 2);` will be evaluated as `((((2*i)%5) * 4) + ((j-3)/(k+2)))` which is 0
  - Whereas `result = 2 * i % (5 * 4) + (j - 3) / (k + 2);` evaluated as 11

Precedence order

Using expressions (contd)

```
int main() {
    int i;

        printf("Enter a two digit number: ");
        scanf("%d", &i);

        printf("Reversed number is: %d\n", i%10*10 + i/10);
}
```

Precedence order

Evaluation of Expressions

- All expressions in parenthesis must be evaluated separately, and inside out.
- The operator precedence rules for operators in same subexpression:
  - Unary + and - are evaluated first
  - *, /, % evaluated next
- Associtivity rule
  - Unary operators in same subexpressions and at same precedence level (such as +, - or *, /) are evaluated right to left.
  - Binary operators in same subexpressions and at same precedence level are evaluated left to right.

Precedence order

Side Effects & Unexpected Behaviors

- `i = j = k = 0` assigns `k = 0` then `j = 0`, finally `i = 0`,
- Subexpression evaluation may produce unexpected results, eg.,
  `a = 5; c = (b = a + 5) - (a = 1);`
  Either `c = 9` or `c = 5` depending on which subexpression is evaluated first.
- Consider `i = 2; j = i * i++;` may give `j = 6` or `j = 4`
- Expression `int i = 1; i += 2;` is different from
  `int i = 1; i += i++ + i++;`, (`i` may be incremented twice).
- **Avoid writing expressions which modify variable within the expression itself**

Common Problems

Problem 1

Suppose an object is thrown up with initial velocity of 50m/sec. How high the object will rise and what time does it take to reach the highest point.

Problem 2

Two persons are standing apart by 1m, each has a mass of 50kg. Let $G = 6.67 \times 10^{-11} Nm^2/kg^2$ and $r_{earth} = 6.64 \times 10^6 m$. Determine the force of F gravitation between P1 and P2. How many times should F be multiplied to get the force of gravitation between the Earth and P1.

Problem 3

How many molecules of H_2O are present in 1 gm of snowflakes? Avogadro number $= 6.022 \times 10^{23}$, and atomic mass of H $= 1.01$ and that of O $= 15.9994$.

Common Problems

Example 8

```
#include <stdio.h>
int main() {
    double g = -10.0;
    double u, t, h;

    printf("Enter initial velocity: ");
    scanf("%lf", &u);
    t = -u/g;                       // since v = 0, t = -u/g
    h = (u + 0.5 * g * t) * t; // d = ut + (1/2) ft^2
    printf("height = %.3f\n", h);
}
```

Common Problems

Example 9

```
#indclude <stdio.h>
int main() {
    double m1, m2, f1, f2, d;
    double G = 6.67e-11, M = 6.0e24, R = 6.4e06;

    printf("Enter masses of two persons: ");
    scanf("%lf %lf", &m1, &m2);
    printf("Enter distance between two persons: ");
    scanf("%lf", &d);

    f1 = (G * m1 * m2) / (d*d);
    f2 = (G * M * m1) / (R * R);

    printf("f2 = %g is %g times of f1 = %g\n",f2, f2/f1, f1);
}
```

Common Problems

Example 10

```
#include <stdio.h>
#define AVOGADRO 6.022e23
int main() {
    double w, mole_mass, mole_val, molecules;

    printf("Enter weight of substance: ");
    scanf("%lf", &w);

    mole_mass = 1.01 * 2.0 + 15.9994;
    mole_val = w/mole_mass;
    molecules = mole_val * AVOGADRO;

    printf("Molecules in %.2f gm of snowflakes = %g\n", w, molecule);
}
```