

Lecture 3

Structure of A C Program

1

Elements of a program

- **Literals** → fixed data written into a program
- **Variables & constants** → placeholders (in memory) for pieces of data
- **Types** → sets of possible values for data
- **Expressions** → combinations of operands (such as variables or even "smaller" expressions) and operators. They compute new values from old ones.
- **Assignments** → used to store values into variables
- **Statements** → "instructions". In C, any expression followed by a semicolon is a statement

2

Elements of a program

- **Control-flow constructs** → constructs that allow statements or groups of statements to be executed only when certain conditions hold or to be executed more than once.
- **Functions** → named blocks of statements that perform a well-defined operation.
- **Libraries** → collections of functions.

Figure 3-13: Type of statements

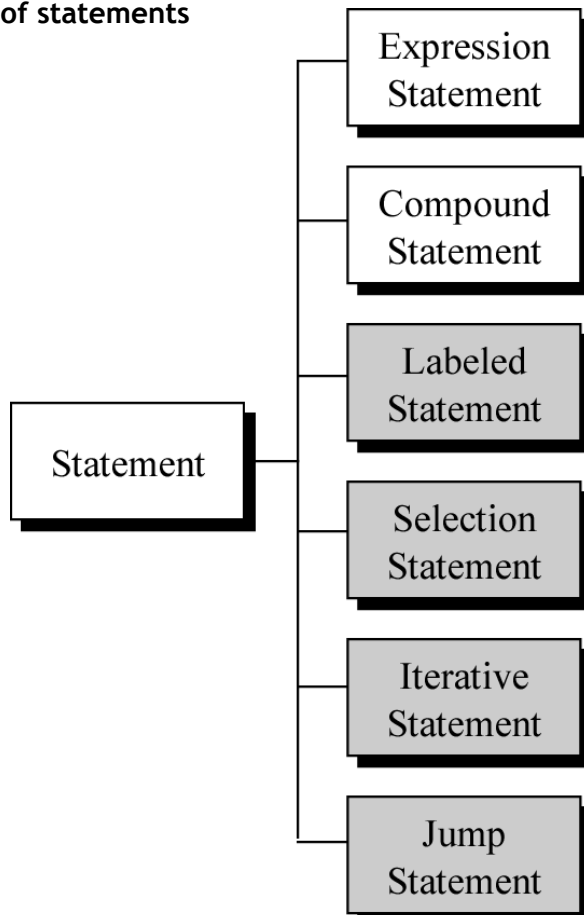


Figure 3-2: Primary expression

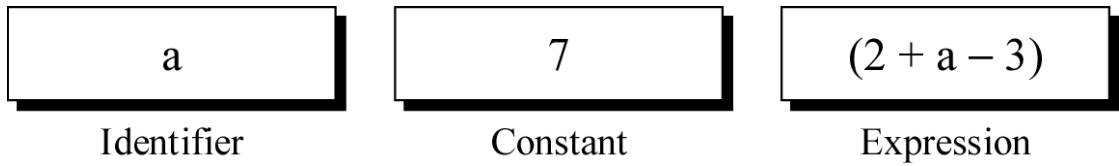


Figure 3-3: Binary expression

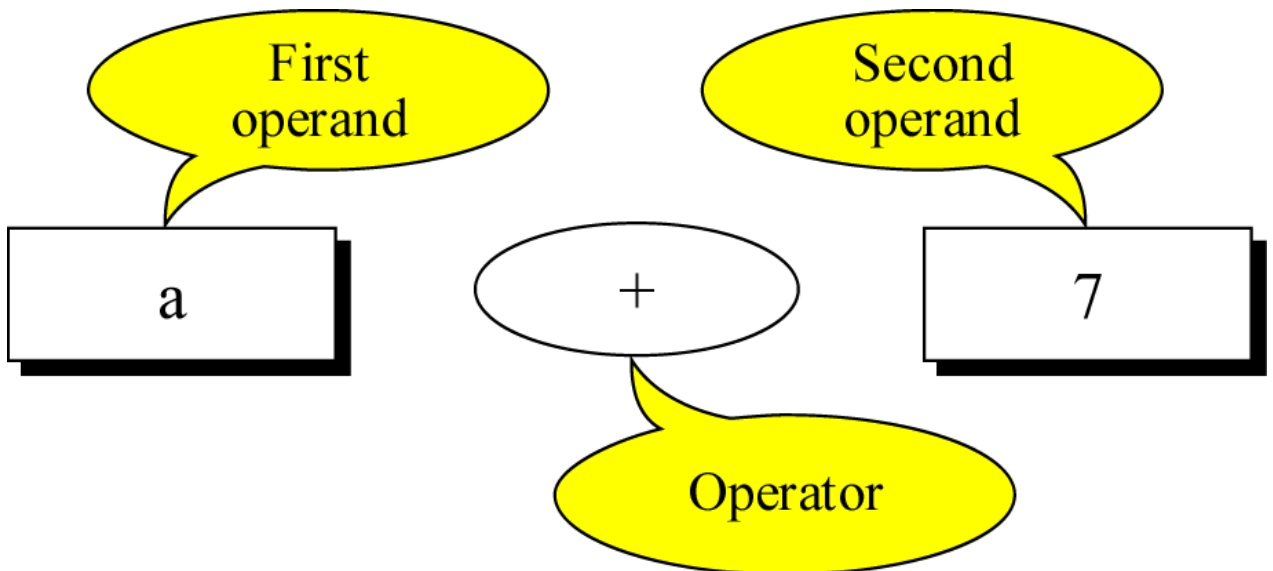


Figure 3-4: Assignment Expression

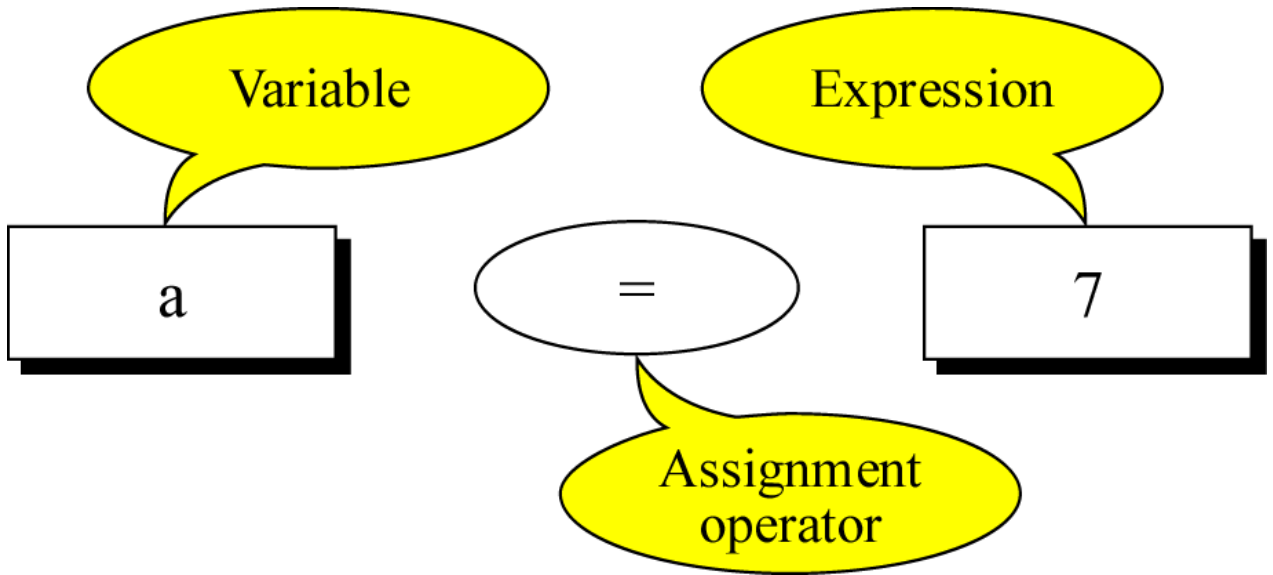


Figure 3-5: Postfix Expression

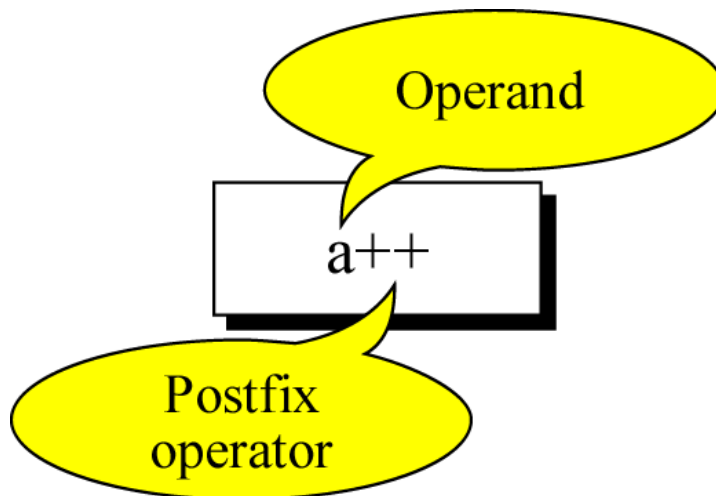


Figure 3-6: Result of postfix a++

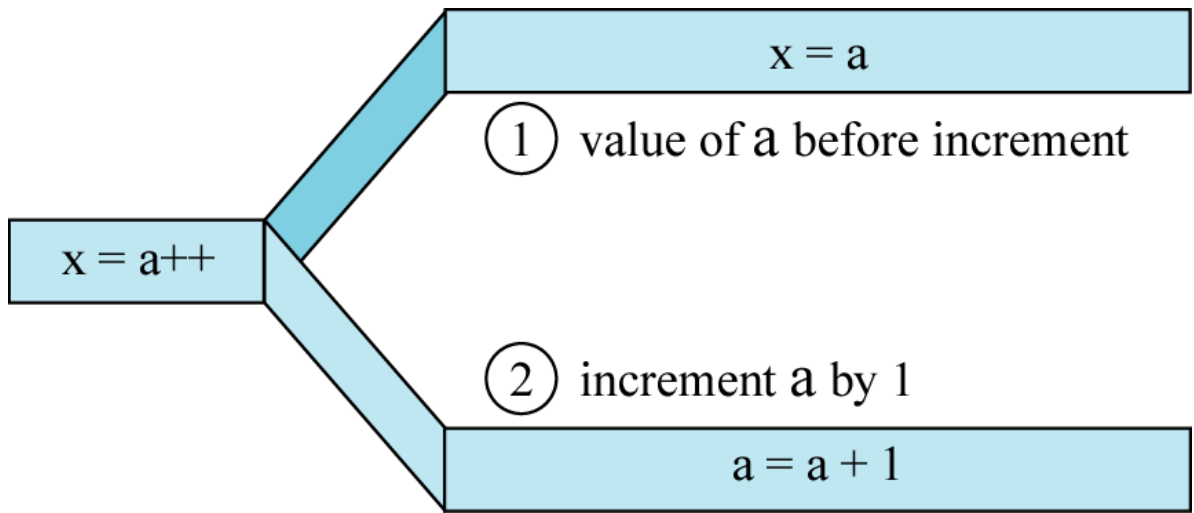


Figure 3-7: Prefix expression

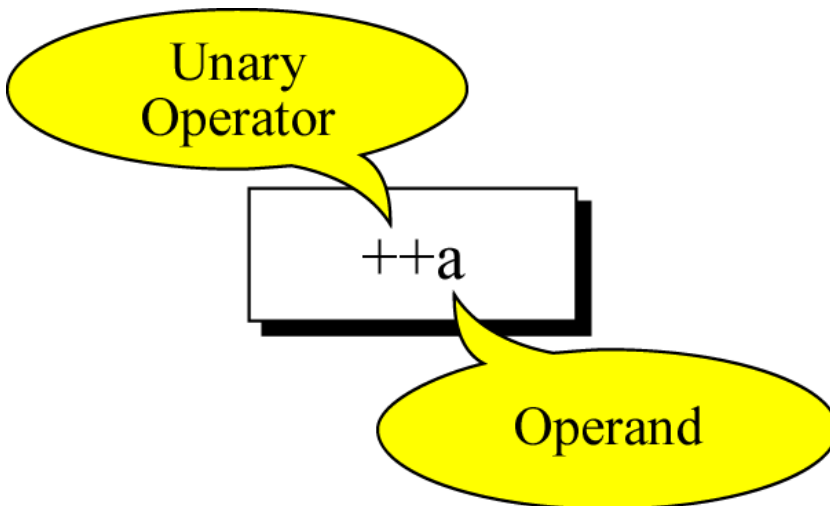
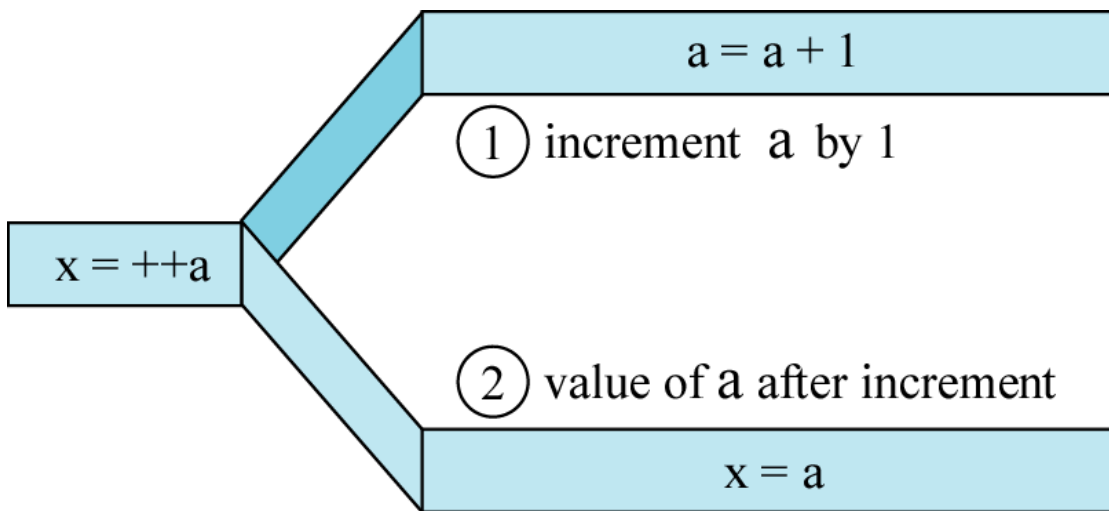


Figure 3-8: Result of prefix ++a



11

Operators

- Operators can be classified according to
 - the type of their operands and of their output
 - Arithmetic
 - Relational
 - Logical
 - Bitwise
 - the number of their operands
 - Unary (one operand)
 - Binary (two operands)

12

Assignment operator: =

- Binary operator used to assign a value to a variable.
- Its left operand is the destination variable
- Its right operand is an expression.

```
int var;  
var = 10;
```



13

Arithmetic operators

- They operate on numbers and the result is a number.
- The type of the result depends on the types of the operands.
- If the types of the operands differ (e.g. an integer added to a floating point number), one is "promoted" to other.
 - The "smaller" type is promoted to the "larger" one.
char → int → float → double

14

Arithmetic operators: +, *

- + is the addition operator
- * is the multiplication operator
- They are both binary

Arithmetic operator: -

- This operator has two meanings:
 - subtraction operator (binary)

e.g. $31 - 2$

- negation operator (unary)

e.g. -10

Arithmetic operator: /

- Division operator
- CAREFUL! The result of integer division is an integer:

e.g. `5 / 2` is `2`, not `2.5`

17

Arithmetic operator: %

- The modulus (remainder) operator.
- It computes the remainder after the first operand is divided by the second

e.g. `5 % 2` is `1`, `6 % 2` is `0`

- It is useful for making cycles of numbers:
 - For an int variable x :

```
if x is: 0 1 2 3 4 5 6 7 8 9 ...
(x%4) is: 0 1 2 3 0 1 2 3 0 1 ...
```

18

Relational operators

- These perform comparisons and the result is what is called a boolean: a value TRUE or FALSE
- FALSE is represented by 0; anything else is TRUE
- The relational operators are:
 - < (less than)
 - <= (less than or equal to)
 - > (greater than)
 - >= (greater than or equal to)
 - == (equal to)
 - != (not equal to)

Logical operators

- These have boolean operands and the result is also a boolean.
- The basic boolean operators are:
 - && (logical AND)
 - || (logical OR)
 - ! (logical NOT) -- unary

Special assignment operators

- write `a += b;` instead of `a = a + b;`
- write `a -= b;` instead of `a = a - b;`
- write `a *= b;` instead of `a = a * b;`
- write `a /= b;` instead of `a = a / b;`
- write `a %= b;` instead of `a = a % b;`

21

Special assignment operators

- Increment, decrement operators: `++`, `--`
 - Instead of `a = a + 1` you can write `a++` or `++a`
 - Instead of `a = a - 1` you can write `a--` or `--a`
- What is the difference?

post-increment

```
num = 10;  
ans = num++;
```

First assign num to ans,
then increment num.

In the end,

num is 11
ans is 10

pre-increment

```
num = 10;  
ans = ++num;
```

First increment num,
then assign num to ans.

In the end,

num is 11
ans is 11

22

Precedence & associativity

- How would you evaluate the expression
 $17 - 8 * 2$?

Is it $17 - (8 * 2)$
or $(17 - 8) * 2$?

- These two forms give different results.
- We need rules!

23

Precedence & associativity

- When two operators compete for the same operand (e.g. in $17 - 8 * 2$ the operators $-$ and $*$ compete for 8) the rules of precedence specify which operator wins.
 - The operator with the higher precedence wins
- If both competing operators have the same precedence, then the rules of associativity determine the winner.

24

Precedence & associativity

!	Unary -		
*	/	%	
+	-		
<	<=	>=	>
=	=	!=	
&&			
=			

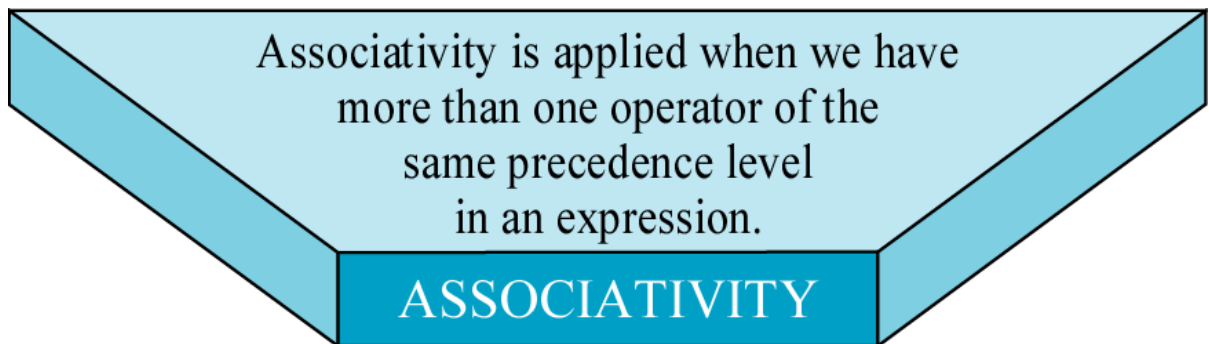
↑ higher precedence

Associativity: execute left-to-right
(**except** for = and unary -)

↓ lower precedence

25

Figure 3-9: Associativity



26

Figure 3-10: Left associativity

Example:

$$3 * 8 / 4 \% 4 * 5$$

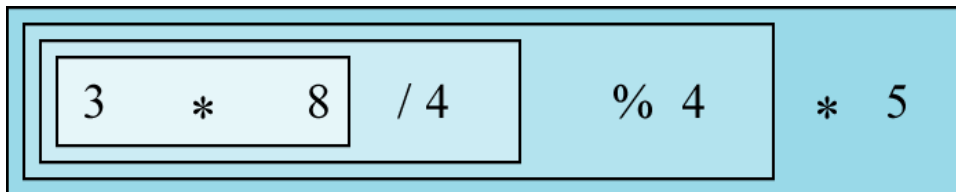
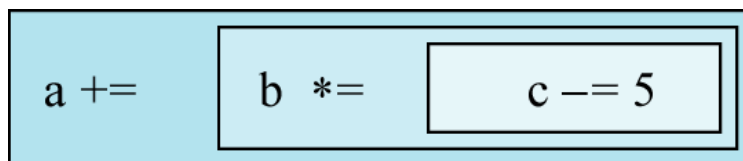


Figure 3-11: Right Associativity

Example:

$$a += b *= c -= 5$$



Precedence & associativity

- Examples:

$$X = 17 - 2 * 8$$

$$\text{Ans: } X = 17 - (2 * 8), X = 1$$

$$Y = 17 - 2 - 8$$

$$\text{Ans: } Y = (17 - 2) - 8, Y = 7$$

$$Z = 10 + 9 * ((8 + 7) \% 6) + 5 * 4 \% 3 * 2 + 1 ?$$

Not sure? Confused? then use parentheses in your code!