Algebra Foundations

Why This Matters

Algebra is the language of *generalization*. Instead of solving "3 + 5 = ?" one at a time, algebra lets you solve "x + 5 = 8, what is x?" for *any* similar problem.

If you've ever written a function with parameters, you've done algebra:

```
function calculate(x, y) {
  return x + y;  // x and y are variables (algebra!)
}
```

Algebra is about:

- Using **variables** as placeholders
- Writing **equations** that describe relationships
- **Solving** for unknowns systematically

The Big Picture: What Problem Does Algebra Solve?

Arithmetic: Specific calculations

```
5 + 3 = 8
10 × 2 = 20
```

Algebra: General patterns and unknowns

```
x + 3 = 8  (find x)
2y = 20    (find y)
a + b = c  (relationship between a, b, c)
```

Algebra lets you:

- Describe patterns
- Work backwards from results to causes
- Solve for unknowns
- Express general rules

---

1. Variables: Placeholders for Values

What They Are

A **variable** is a symbol (usually a letter) that stands for a number you don't know yet, or a number that can change.

Common variable names: x, y, z, a, b, c, n, t

Programming Analogy:

```
let x = 5;        // x is a variable
let y = x + 3;    // y depends on x
```

In math, variables work the same way—they're containers for values.

Variables as Function Parameters

When you write:

```
function double(n) {
  return n * 2;
}
```

`n` is a variable. It could be 5, 10, or 100. Same in algebra:

```
f(n) = 2n
```

This means "whatever n is, double it."

Constants vs Variables

- **Variable**: Can change (x, y, n)
- **Constant**: Fixed value (5, π, -2)

Example:

```
y = 2x + 3
```

- `x` and `y` are variables
- `2` and `3` are constants

Multiple Variables

You can have more than one:

```
area = length × width
A = l × w
```

Both `l` and `w` are variables. `A` (area) depends on them.

---

2. Expressions vs Equations

Expression

A combination of numbers, variables, and operations **without an equals sign**.

Examples:

```
3x + 5
2a - 7
x² + 2x + 1
```

Think of expressions as **incomplete statements** or **formulas**. They evaluate to a value but don't claim anything.

Programming Analogy:

```
const expression = x * 2 + 5;  // just a value, not a comparison
```

Equation

Two expressions set equal to each other **with an equals sign**.

Examples:

```
3x + 5 = 14
2a - 7 = 3
y = x² + 2x + 1
```

Equations make a **claim**: "These two things are equal."

Programming Analogy:

```
const equation = (3*x + 5 === 14);  // true or false
```

Key Difference

- **Expression**: "What is 3x + 5 when x = 2?" → Evaluate it
- **Equation**: "When does 3x + 5 = 14?" → Solve it

---

3. Evaluating Expressions

What It Means

Substitute a value for the variable and **compute** the result.

Example:

```
Expression: 3x + 5
If x = 2, then:
3(2) + 5 = 6 + 5 = 11
```

Step by Step

```
Evaluate: 2x² - 3x + 1  when x = 4

Step 1: Substitute x = 4
2(4)² - 3(4) + 1

Step 2: Exponents first
2(16) - 3(4) + 1

Step 3: Multiply
```

```
32 - 12 + 1

Step 4: Add/subtract left to right
20 + 1 = 21

Answer: 21
```

Programming Analogy:

```
function evaluate(x) {
  return 2*x**2 - 3*x + 1;
}

console.log(evaluate(4));  // 21
```

---

4. Order of Operations (Why It Exists)

The Problem

What does `2 + 3 × 4` equal?

- If you go left to right: (2 + 3) × 4 = 20 ❌
- If you do multiplication first: 2 + (3 × 4) = 14 ✓

We need rules so everyone gets the same answer.

PEMDAS (or BODMAS)

The order you must follow:

1. **P**arentheses (or **B**rackets)
2. **E**xponents (or **O**rders)
3. **M**ultiplication and **D**ivision (left to right)
4. **A**ddition and **S**ubtraction (left to right)

Why This Order?

It's a convention, but it makes sense:

- **Parentheses**: Explicit grouping (you decide)
- **Exponents**: Repeated multiplication (compact)
- **Multiplication/Division**: More "binding" than addition
- **Addition/Subtraction**: Least binding

Examples

Example 1

```
2 + 3 × 4

Step 1: Multiply first
2 + 12
```

```
Step 2: Add
14
```

Example 2

```
(2 + 3) × 4

Step 1: Parentheses first
5 × 4

Step 2: Multiply
20
```

Example 3

```
10 - 2 × 3 + 4

Step 1: Multiply
10 - 6 + 4

Step 2: Left to right
4 + 4 = 8
```

Example 4

```
2 + 3² × (4 - 1)

Step 1: Parentheses
2 + 3² × 3

Step 2: Exponent
2 + 9 × 3

Step 3: Multiply
2 + 27

Step 4: Add
29
```

Programming Note

```
console.log(2 + 3 * 4);        // 14 (same rules!)
console.log((2 + 3) * 4);      // 20 (parentheses change it)
```

Programming languages follow the same order of operations.

---

5. Solving Equations: Undoing Operations

What Does "Solve" Mean?

To solve an equation means to find the value(s) of the variable that make the equation true.

Example:

```
x + 5 = 12

What value of x makes this true?
x = 7  (because 7 + 5 = 12)
```

The Golden Rule: Do the Same Thing to Both Sides

Whatever you do to one side, do to the other.

Equations are like balanced scales:

```
    x + 5   =   12
   ┌─────┐   ┌─────┐
   │     │   │     │
   └──┬──┘   └──┬──┘
      │         │
    ──┴─────────┴──
```

If you add/subtract/multiply/divide one side, do it to the other to keep the balance.

Strategy: Undo Operations in Reverse Order

Think of solving equations as **reversing a series of operations**.

If the equation does: x → add 5 → multiply by 2 Then to solve: divide by 2 ← subtract 5 ← x

Programming Analogy:

```
// Building the equation
let result = (x + 5) * 2;

// Solving (undo in reverse)
// result / 2 = x + 5
// (result / 2) - 5 = x
let x = (result / 2) - 5;
```

---

6. Solving Linear Equations

Type 1: Addition/Subtraction

```
x + 5 = 12

Goal: Isolate x (get x alone)

Step 1: Subtract 5 from both sides
x + 5 - 5 = 12 - 5
x = 7
```

```
Check: 7 + 5 = 12 ✓
```

Type 2: Multiplication/Division

```
3x = 15

Goal: Isolate x

Step 1: Divide both sides by 3
3x / 3 = 15 / 3
x = 5

Check: 3(5) = 15 ✓
```

Type 3: Multiple Steps

```
2x + 7 = 15

Step 1: Subtract 7 from both sides
2x = 8

Step 2: Divide both sides by 2
x = 4

Check: 2(4) + 7 = 8 + 7 = 15 ✓
```

Type 4: Variables on Both Sides

```
5x + 3 = 2x + 12

Step 1: Subtract 2x from both sides
3x + 3 = 12

Step 2: Subtract 3 from both sides
3x = 9

Step 3: Divide by 3
x = 3

Check: 5(3) + 3 = 15 + 3 = 18
       2(3) + 12 = 6 + 12 = 18 ✓
```

Type 5: Fractions

```
x/4 = 3

Step 1: Multiply both sides by 4
x = 12
```

```
Check: 12/4 = 3 ✓
```

More complex:

```
(x + 2)/3 = 5

Step 1: Multiply both sides by 3
x + 2 = 15

Step 2: Subtract 2
x = 13

Check: (13 + 2)/3 = 15/3 = 5 ✓
```

---

7. Working with Negative Coefficients

Example 1

```
-x = 5

Step 1: Multiply both sides by -1
x = -5

Check: -(-5) = 5 ✓
```

Example 2

```
-3x + 4 = 10

Step 1: Subtract 4
-3x = 6

Step 2: Divide by -3
x = -2

Check: -3(-2) + 4 = 6 + 4 = 10 ✓
```

Remember: Dividing or multiplying by a negative flips the sign.

---

8. Distributing (The Distributive Property)

What It Means

```
a(b + c) = ab + ac
```

You **distribute** the multiplication over the addition.

Visual

```
3(x + 2) means "3 groups of (x + 2)"

Group 1: x + 2
Group 2: x + 2
Group 3: x + 2
─────────────────
Total: 3x + 6
```

Examples

```
5(x + 3) = 5x + 15
2(3x - 4) = 6x - 8
-3(2x + 1) = -6x - 3
```

Why It's Useful

It lets you simplify expressions and solve equations:

```
3(x + 2) = 21

Step 1: Distribute
3x + 6 = 21

Step 2: Subtract 6
3x = 15

Step 3: Divide by 3
x = 5

Check: 3(5 + 2) = 3(7) = 21 ✓
```

Programming Analogy

```
// Without distributing
const result = 3 * (x + 2);

// Distributed (equivalent)
const result = 3*x + 6;
```

---

9. Combining Like Terms

What Are Like Terms?

Terms with the **same variable and exponent**.

Like terms:

```
3x and 5x    (both have x)
2y² and -7y² (both have y²)
```

NOT like terms:

```
3x and 5y    (different variables)
2x and 2x²   (different exponents)
```

How to Combine

Add or subtract the **coefficients** (numbers in front):

```
3x + 5x = 8x
7y - 2y = 5y
4x² + 3x² = 7x²
```

Example

```
Simplify: 3x + 5 + 2x - 3

Step 1: Group like terms
(3x + 2x) + (5 - 3)

Step 2: Combine
5x + 2
```

Why It Matters

Combining like terms simplifies equations:

```
5x + 3x - 2 = 14

Step 1: Combine like terms
8x - 2 = 14

Step 2: Add 2
8x = 16

Step 3: Divide by 8
x = 2
```

---

10. Inequalities: Ranges, Not Points

What They Are

Inequalities use symbols like `<`, `>`, `≤`, `≥` instead of `=`.

Symbol	Meaning
<	Less than
>	Greater than
≤	Less than or equal to

≥	Greater than or equal to

Examples

```
x > 5     (x is greater than 5)
y ≤ 10    (y is less than or equal to 10)
-3 < z    (z is greater than -3)
```

Visual: Number Line

```
x > 5:
   ────────●═══════→
           5
     (5 not included, everything to the right)

x ≥ 5:
   ────────●═══════→
           5
     (5 IS included, everything to the right)
```

- **Open circle** (○): Not included (<, >)
- **Filled circle** (●): Included (≤, ≥)

Solving Inequalities

Works just like equations... with ONE exception.

Normal Operations

```
x + 3 > 7

Subtract 3:
x > 4
```

THE EXCEPTION: Multiplying/Dividing by Negatives

When you multiply or divide by a negative number, FLIP the inequality sign.

```
-2x > 6

Divide by -2 (and flip the sign):
x < -3
```

Why? Because multiplying by a negative reverses order:

```
5 > 3  (true)
Multiply both by -1:
-5 < -3  (still true, but flipped)
```

Example with Multiple Steps

```
-3x + 4 ≤ 10

Step 1: Subtract 4
-3x ≤ 6

Step 2: Divide by -3 (flip sign!)
x ≥ -2
```

Compound Inequalities

You can have two inequalities at once:

```
1 < x < 5

This means: x is between 1 and 5
(x > 1 AND x < 5)
```

Visual:

```
    •───────○
    1       5
```

Programming Analogy

```
if (x > 5) {
  console.log("x is greater than 5");
}

if (x >= 5 && x <= 10) {
  console.log("x is between 5 and 10 (inclusive)");
}
```

Common Mistakes & Misconceptions

❌ "Variables are always x"

Variables can be any letter: y, z, a, n, t, θ. Choose meaningful names like `time`, `distance`.

❌ "3x means 3 + x"

No. 3x means 3 × x. If you see a number next to a variable, it's multiplication.

❌ "You can't have negative solutions"

Negative solutions are totally valid: x = -5 is a perfectly good answer.

❌ "Both sides of an equation must look the same"

No. They must *equal* the same value, but can look different:

```
2x + 3 = 11  (left and right look different but equal 11 when x=4)
```

❌ "Dividing both sides by the variable"

Be careful:

```
2x = 3x  →  Don't divide by x!

Instead, subtract 3x:
-x = 0
x = 0
```

Dividing by x assumes $x \neq 0$, which might not be true.

❌ "Forgetting to flip inequality when multiplying/dividing by negative"

```
-2x > 6  →  x < -3  (not x > -3)
```

Real-World Examples

Shopping (Linear Equations)

```
You have $20. Apples cost $2 each. How many can you buy?

2x = 20
x = 10 apples
```

Speed and Distance

```
Distance = Speed × Time
d = st

If you travel at 60 mph for 2.5 hours:
d = 60 × 2.5 = 150 miles
```

Temperature Conversion

```
Fahrenheit to Celsius:
C = (F - 32) × 5/9

If F = 68:
C = (68 - 32) × 5/9 = 36 × 5/9 = 20°C
```

Programming: Loop Conditions

```
for (let i = 0; i < 10; i++) {  // i < 10 is an inequality
  console.log(i);
}
```

Budget Constraints

```
You want to spend no more than $100:
cost ≤ 100
```

Tiny Practice

Solve these equations:

1. `x + 7 = 15`
2. `3x = 27`
3. `2x - 5 = 13`
4. `5x + 3 = 2x + 12`
5. `-4x = 20`
6. `3(x + 2) = 18`

Solve these inequalities:

7. `x + 5 > 12`
8. `-2x ≤ 10`
9. `3x - 1 < 8`

Simplify:

10. `5x + 3x - 2`
11. `2(3x + 4) - 5`

▶ Answers

Summary Cheat Sheet

Key Concepts

Concept	Definition	Example
Variable	Placeholder for a value	x, y, z
Expression	Numbers, variables, operations (no =)	3x + 5
Equation	Two expressions set equal	3x + 5 = 14
Solving	Finding values that make equation true	x = 3

Order of Operations: PEMDAS

1. **P**arentheses
2. **E**xponents
3. **M**ultiply/**D**ivide (left to right)
4. **A**dd/**S**ubtract (left to right)

Solving Equations

1. **Simplify** both sides (distribute, combine like terms)
2. **Isolate** the variable (undo operations in reverse)

3. **Do the same** to both sides
4. **Check** your answer

Special Rules

- **Distributive Property**: a(b + c) = ab + ac
- **Combining Like Terms**: 3x + 5x = 8x
- **Inequality Flip**: When multiplying/dividing by negative, flip the sign

Programming Connections

```
// Variables
let x = 5;

// Expressions
let result = 2*x + 3;

// Equations (checking)
if (2*x + 3 === 13) { /* true when x=5 */ }

// Inequalities
if (x > 3) { /* condition */ }
```

Next Steps

You now understand how to work with variables, expressions, and equations. You can solve for unknowns and express general relationships.

Next, we'll explore **Ratios, Proportions, and Percentages**—how to compare quantities and scale values.

Continue to: 02-ratios-proportions-percentages.md