Lecture 02: Operator, if-else, loop in C++

1. Operators

A. Arithmetic Operators

Used for math calculations.

```
int a = 10, b = 3;

cout << a + b;   // 13 (addition)
cout << a - b;   // 7  (subtraction)
cout << a * b;   // 30 (multiplication)
cout << a / b;   // 3  (division)
cout << a % b;   // 1  (remainder/modulo)
```

Example:

```
int marks1 = 80, marks2 = 90;
int total = marks1 + marks2;
int average = total / 2;

cout << "Total: " << total << endl;      // 170
cout << "Average: " << average << endl;  // 85
```

B. Assignment Operators

Used to assign values.

```
int x = 10;

x = 5;      // x becomes 5
x += 3;     // x = x + 3  → x becomes 8
x -= 2;     // x = x - 2  → x becomes 6
x *= 4;     // x = x * 4  → x becomes 24
x /= 2;     // x = x / 2  → x becomes 12
```

C. Comparison Operators

Used to compare values. Result is `true` or `false`.

```
int a = 10, b = 5;

a == b    // false (is equal?)
a != b    // true  (is not equal?)
a > b     // true  (is greater?)
a < b     // false (is smaller?)
a >= b    // true  (is greater or equal?)
a <= b    // false (is smaller or equal?)
```

D. Logical Operators

Combine multiple conditions.

```
// AND (&&) - Both must be true
true && true    // true
true && false   // false

// OR (||) - At least one must be true
true || false   // true
false || false  // false

// NOT (!) - Reverses
!true    // false
!false   // true
```

Example:

```
int age = 20;
bool hasLicense = true;

// Can drive if age >= 18 AND has license
if(age >= 18 && hasLicense) {
    cout << "Can drive!";
}
```

E. Increment/Decrement

Quick way to add or subtract 1.

```
int x = 5;

x++;    // x becomes 6 (same as x = x + 1)
x--;    // x becomes 5 (same as x = x - 1)
```

Example:

```
int score = 0;
score++;  // score becomes 1
score++;  // score becomes 2
score++;  // score becomes 3
```

---

4. Input and Output

Output (cout)

Print to screen.

```
cout << "Hello World";
cout << "Age: " << 25;
cout << "Name: " << name << endl;  // endl = new line
```

Input (cin)

Take input from user.

```
int age;
cout << "Enter your age: ";
cin >> age;

string name;
cout << "Enter your name: ";
cin >> name;
```

Complete Example:

```
#include <iostream>
using namespace std;

int main() {
    string name;
    int age;

    cout << "Enter your name: ";
    cin >> name;

    cout << "Enter your age: ";
    cin >> age;

    cout << "\nHello " << name << "!" << endl;
    cout << "You are " << age << " years old." << endl;

    return 0;
}
```

2. If-Else Statements

Basic If Statement

```
if(condition) {
    // code runs if condition is true
}
```

Example:

```
int age = 20;

if(age >= 18) {
    cout << "You are an adult";
}
```

If-Else Statement

```
if(condition) {
    // code runs if condition is true
} else {
    // code runs if condition is false
}
```

Example:

```
int age = 15;

if(age >= 18) {
    cout << "You are an adult";
} else {
    cout << "You are a minor";
}
```

If-Else If-Else Statement

```
if(condition1) {
    // runs if condition1 is true
} else if(condition2) {
    // runs if condition2 is true
} else {
    // runs if all conditions are false
}
```

Example:

```
int marks = 85;

if(marks >= 90) {
    cout << "Grade: A";
} else if(marks >= 80) {
    cout << "Grade: B";
} else if(marks >= 70) {
    cout << "Grade: C";
} else if(marks >= 60) {
    cout << "Grade: D";
} else {
    cout << "Grade: F";
}
```

Nested If (If inside If)

```
int age = 20;
bool hasLicense = true;

if(age >= 18) {
    if(hasLicense) {
        cout << "You can drive";
    } else {
        cout << "You need a license";
    }
} else {
    cout << "You are too young";
}
```

Practice Problems

Problem 1: Even or Odd

```
#include <iostream>
using namespace std;

int main() {
    int num;
    cout << "Enter a number: ";
    cin >> num;

    if(num % 2 == 0) {
        cout << num << " is even";
    } else {
        cout << num << " is odd";
    }

    return 0;
}
```

Problem 2: Maximum of Two Numbers

```
#include <iostream>
using namespace std;

int main() {
    int a, b;

    cout << "Enter first number: ";
    cin >> a;
```

```cpp
    cout << "Enter second number: ";
    cin >> b;

    if(a > b) {
        cout << a << " is maximum";
    } else {
        cout << b << " is maximum";
    }

    return 0;
}
```

Problem 3: Positive, Negative, or Zero

```cpp
#include <iostream>
using namespace std;

int main() {
    int num;
    cout << "Enter a number: ";
    cin >> num;

    if(num > 0) {
        cout << "Positive";
    } else if(num < 0) {
        cout << "Negative";
    } else {
        cout << "Zero";
    }

    return 0;
}
```

Problem 4: Calculator

```
#include <iostream>
using namespace std;

int main() {
    double num1, num2;
    char op;

    cout << "Enter first number: ";
    cin >> num1;

    cout << "Enter operator (+, -, *, /): ";
    cin >> op;

    cout << "Enter second number: ";
    cin >> num2;

    if(op == '+') {
        cout << "Result: " << (num1 + num2);
    } else if(op == '-') {
        cout << "Result: " << (num1 - num2);
    } else if(op == '*') {
        cout << "Result: " << (num1 * num2);
    } else if(op == '/') {
        if(num2 != 0) {
            cout << "Result: " << (num1 / num2);
        } else {
            cout << "Cannot divide by zero!";
        }
    } else {
        cout << "Invalid operator!";
    }

    return 0;
}
```

Problem 5: Grade Calculator

```
#include <iostream>
using namespace std;

int main() {
    int marks;

    cout << "Enter your marks (0-100): ";
    cin >> marks;

    if(marks > 100 || marks < 0) {
        cout << "Invalid marks!";
    } else if(marks >= 90) {
        cout << "Grade: A (Excellent)";
    } else if(marks >= 80) {
        cout << "Grade: B (Very Good)";
    } else if(marks >= 70) {
        cout << "Grade: C (Good)";
    } else if(marks >= 60) {
        cout << "Grade: D (Satisfactory)";
    } else {
        cout << "Grade: F (Fail)";
    }

    return 0;
}
```

Summary Cheat Sheet

Variables

```
int age = 25;
double price = 99.99;
char grade = 'A';
bool flag = true;
string name = "Rohit";
```

Operators

```
+  -  *  /  %          // Math
=  +=  -=  *=  /=      // Assignment
==  !=  >  <  >=  <= // Comparison
&&  ||  !              // Logical
++  --                 // Increment/Decrement
```

Input/Output

```
cout << "Text";        // Output
cin >> variable;       // Input
getline(cin, str);     // Input with spaces
```

If-Else

```
if(condition) {
    // code
} else if(condition) {
    // code
} else {
    // code
}
```

Switch Statement

What is Switch?

Alternative to multiple `if-else if` statements. Better for checking **one variable** against many values.

Syntax:

```
switch(variable) {
    case value1:
        // code
        break;
    case value2:
        // code
        break;
    default:
        // code if no match
}
```

Example 1: Days of Week

```
int day = 3;

switch(day) {
    case 1:
        cout << "Monday" << endl;
        break;
    case 2:
        cout << "Tuesday" << endl;
        break;
    case 3:
        cout << "Wednesday" << endl;
        break;
    case 4:
        cout << "Thursday" << endl;
        break;
    case 5:
        cout << "Friday" << endl;
        break;
    case 6:
        cout << "Saturday" << endl;
        break;
    case 7:
        cout << "Sunday" << endl;
        break;
    default:
        cout << "Invalid day" << endl;
}
```

Output:

```
Wednesday
```

Example 2: Calculator

```
char op;
double a, b;

cout << "Enter operator (+, -, *, /): ";
cin >> op;

cout << "Enter two numbers: ";
cin >> a >> b;

switch(op) {
    case '+':
        cout << "Result: " << (a + b) << endl;
        break;
    case '-':
        cout << "Result: " << (a - b) << endl;
        break;
    case '*':
        cout << "Result: " << (a * b) << endl;
        break;
    case '/':
        if(b != 0) {
            cout << "Result: " << (a / b) << endl;
        } else {
            cout << "Cannot divide by zero!" << endl;
        }
        break;
    default:
        cout << "Invalid operator" << endl;
}
```

Important: `break` Statement

Without `break`, code "falls through":

```
int num = 2;

switch(num) {
    case 1:
        cout << "One" << endl;
    case 2:
        cout << "Two" << endl;  // Prints this
    case 3:
        cout << "Three" << endl; // Also prints this!
    default:
        cout << "Other" << endl; // And this too!
}
```

Output:

```
Two
Three
```

```
Other
```

With `break`:

```
int num = 2;

switch(num) {
    case 1:
        cout << "One" << endl;
        break;
    case 2:
        cout << "Two" << endl;
        break;  // Stops here
    case 3:
        cout << "Three" << endl;
        break;
    default:
        cout << "Other" << endl;
}
```

Output:

```
Two
```

Part 3: Loops

What are Loops?

Repeat code multiple times without writing it again.

3 Types:

1. `for` loop
2. `while` loop
3. `do-while` loop

3.1 `for` Loop

Best when you know HOW MANY times to repeat.

Syntax:

```
for(initialization; condition; update) {
    // code to repeat
}
```

Example: Print 1 to 5

```
for(int i = 1; i <= 5; i++) {
    cout << i << endl;
}
```

Output:

```
1
2
3
4
5
```

How `for` Loop Works

```
for(int i = 1; i <= 5; i++) {
    cout << i << endl;
}
```

Step by step:

```
Step 1: i = 1 (initialization)
Step 2: Check i <= 5? YES → Print 1
Step 3: i++ → i becomes 2
Step 4: Check i <= 5? YES → Print 2
Step 5: i++ → i becomes 3
Step 6: Check i <= 5? YES → Print 3
Step 7: i++ → i becomes 4
Step 8: Check i <= 5? YES → Print 4
Step 9: i++ → i becomes 5
Step 10: Check i <= 5? YES → Print 5
Step 11: i++ → i becomes 6
Step 12: Check i <= 5? NO → Stop
```

Example 1: Print Even Numbers

```
for(int i = 2; i <= 10; i += 2) {
    cout << i << " ";
}
```

Output:

```
2 4 6 8 10
```

Example 2: Sum of First 10 Numbers

```
int sum = 0;

for(int i = 1; i <= 10; i++) {
    sum = sum + i;
}

cout << "Sum: " << sum << endl;
```

Output:

```
Sum: 55
```

Example 3: Multiplication Table

```
int num = 5;

cout << "Multiplication table of " << num << endl;

for(int i = 1; i <= 10; i++) {
    cout << num << " x " << i << " = " << (num * i) << endl;
}
```

Output:

```
Multiplication table of 5
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
...
5 x 10 = 50
```

Example 4: Reverse Counting

```
for(int i = 10; i >= 1; i--) {
    cout << i << " ";
}
```

Output:

```
10 9 8 7 6 5 4 3 2 1
```

3.2 `while` Loop

Best when you DON'T know how many times to repeat.

Syntax:

```
while(condition) {
    // code to repeat
}
```

Example: Print 1 to 5

```
int i = 1;

while(i <= 5) {
    cout << i << endl;
    i++;
}
```

Output:

```
1
2
3
4
5
```

Example 1: User Input Until Correct

```
int password;
int correctPassword = 1234;

cout << "Enter password: ";
cin >> password;

while(password != correctPassword) {
    cout << "Wrong! Try again: ";
    cin >> password;
}

cout << "Access granted!" << endl;
```

Example 2: Sum Until User Enters 0

```
int num;
int sum = 0;

cout << "Enter numbers (0 to stop):" << endl;

while(true) {
    cin >> num;

    if(num == 0) {
        break;  // Exit loop
    }

    sum += num;
}

cout << "Total sum: " << sum << endl;
```

3.3 `do-while` Loop

Runs AT LEAST ONCE, then checks condition.

Syntax:

```
do {
    // code to repeat
} while(condition);
```

Example:

```
int i = 1;

do {
    cout << i << endl;
    i++;
} while(i <= 5);
```

Output:

```
1
2
3
4
5
```

Difference: `while` vs `do-while`

`while` loop:

```
int i = 10;

while(i <= 5) {  // Condition false, never runs
    cout << i << endl;
}
```

Output: (nothing)

`do-while` loop:

```
int i = 10;

do {
    cout << i << endl;  // Runs once
} while(i <= 5);
```

Output:

```
10
```

Part 4: Loop Control Statements

4.1 `break` - Exit Loop Immediately

```
for(int i = 1; i <= 10; i++) {
    if(i == 5) {
        break;  // Stop when i is 5
    }
    cout << i << " ";
}
```

Output:

```
1 2 3 4
```

4.2 `continue` - Skip Current Iteration

```
for(int i = 1; i <= 5; i++) {
    if(i == 3) {
        continue;  // Skip 3
    }
    cout << i << " ";
}
```

Output:

```
1 2 4 5
```

Example: Skip Even Numbers

```
for(int i = 1; i <= 10; i++) {
    if(i % 2 == 0) {
        continue;  // Skip even numbers
    }
    cout << i << " ";
}
```

Output:

```
1 3 5 7 9
```

Pattern Printing & Type Conversion

Part 1: Pattern Printing (Complete Guide)

Pattern 1: Square Pattern

```
// Print 5x5 square of stars
*****
*****
*****
*****
*****
```

Code:

```
for(int i = 1; i <= 5; i++) {
    for(int j = 1; j <= 5; j++) {
        cout << "*";
    }
    cout << endl;
}
```

Pattern 2: Right Triangle (Increasing)

```
*
**
***
****
```

```
****
*****
```

Code:

```
for(int i = 1; i <= 5; i++) {
    for(int j = 1; j <= i; j++) {
        cout << "*";
    }
    cout << endl;
}
```

How it works:

```
i=1: Print 1 star
i=2: Print 2 stars
i=3: Print 3 stars
i=4: Print 4 stars
i=5: Print 5 stars
```

Pattern 3: Right Triangle (Decreasing)

```
*****
****
***
**
*
```

Code:

```
for(int i = 5; i >= 1; i--) {
    for(int j = 1; j <= i; j++) {
        cout << "*";
    }
    cout << endl;
}
```

Pattern 4: Inverted Right Triangle

```
    *
   **
  ***
 ****
*****
```

Code:

```
for(int i = 1; i <= 5; i++) {
    // Print spaces
    for(int j = 1; j <= 5-i; j++) {
        cout << " ";
    }
    // Print stars
    for(int j = 1; j <= i; j++) {
        cout << "*";
    }
    cout << endl;
}
```

Logic:

```
Row 1: 4 spaces, 1 star
Row 2: 3 spaces, 2 stars
Row 3: 2 spaces, 3 stars
Row 4: 1 space,  4 stars
Row 5: 0 spaces, 5 stars
```

Pattern 5: Pyramid

```
    *
   ***
  *****
 *******
*********
```

Code:

```
int n = 5;

for(int i = 1; i <= n; i++) {
    // Print spaces
    for(int j = 1; j <= n-i; j++) {
        cout << " ";
    }
    // Print stars
    for(int j = 1; j <= 2*i-1; j++) {
        cout << "*";
    }
    cout << endl;
}
```

Logic:

```
Row 1: 4 spaces, 1 star  (2×1-1 = 1)
Row 2: 3 spaces, 3 stars (2×2-1 = 3)
Row 3: 2 spaces, 5 stars (2×3-1 = 5)
Row 4: 1 space,  7 stars (2×4-1 = 7)
Row 5: 0 spaces, 9 stars (2×5-1 = 9)
```

Pattern 6: Inverted Pyramid

```
*********
 *******
  *****
   ***
    *
```

Code:

```
int n = 5;

for(int i = n; i >= 1; i--) {
    // Print spaces
    for(int j = 1; j <= n-i; j++) {
        cout << " ";
    }
    // Print stars
    for(int j = 1; j <= 2*i-1; j++) {
        cout << "*";
    }
    cout << endl;
}
```

Pattern 7: Diamond

```
    *
   ***
  *****
 *******
*********
 *******
  *****
   ***
    *
```

Code:

```
int n = 5;

// Upper half (pyramid)
for(int i = 1; i <= n; i++) {
    for(int j = 1; j <= n-i; j++) {
        cout << " ";
    }
    for(int j = 1; j <= 2*i-1; j++) {
        cout << "*";
    }
    cout << endl;
}

// Lower half (inverted pyramid)
for(int i = n-1; i >= 1; i--) {
    for(int j = 1; j <= n-i; j++) {
        cout << " ";
    }
    for(int j = 1; j <= 2*i-1; j++) {
        cout << "*";
    }
    cout << endl;
}
```

Pattern 8: Hollow Square

```
*****
*   *
*   *
*   *
*****
```

Code:

```
int n = 5;

for(int i = 1; i <= n; i++) {
    for(int j = 1; j <= n; j++) {
        if(i == 1 || i == n || j == 1 || j == n) {
            cout << "*";
        } else {
            cout << " ";
        }
    }
    cout << endl;
}
```

Pattern 9: Number Triangle

```
1
12
123
1234
12345
```

Code:

```
for(int i = 1; i <= 5; i++) {
    for(int j = 1; j <= i; j++) {
        cout << j;
    }
    cout << endl;
}
```

Pattern 10: Number Triangle (Same Number)

```
1
22
333
4444
55555
```

Code:

```
for(int i = 1; i <= 5; i++) {
    for(int j = 1; j <= i; j++) {
        cout << i;
    }
    cout << endl;
}
```

Pattern 11: Number Triangle (Continuous)

```
1
23
456
78910
```

Code:

```
int num = 1;

for(int i = 1; i <= 4; i++) {
    for(int j = 1; j <= i; j++) {
        cout << num;
        num++;
    }
    cout << endl;
}
```

Pattern 12: Floyd's Triangle

```
1
2 3
4 5 6
7 8 9 10
11 12 13 14 15
```

```

```

Code:

```
int num = 1;

for(int i = 1; i <= 5; i++) {
    for(int j = 1; j <= i; j++) {
        cout << num << " ";
        num++;
    }
    cout << endl;
}
```

Pattern 13: Binary Triangle

```
1
01
101
0101
10101
```

Code:

```
for(int i = 1; i <= 5; i++) {
    for(int j = 1; j <= i; j++) {
        if((i + j) % 2 == 0) {
            cout << "1";
        } else {
            cout << "0";
        }
    }
    cout << endl;
}
```

Pattern 14: Alphabet Triangle

```
A
AB
ABC
ABCD
ABCDE
```

Code:

```
for(int i = 1; i <= 5; i++) {
    char ch = 'A';
    for(int j = 1; j <= i; j++) {
        cout << ch;
        ch++;
    }
    cout << endl;
}
```

Pattern 15: Alphabet Triangle (Repeated)

```
A
BB
CCC
DDDD
```

```
EEEEE
```

Code:

```
char ch = 'A';

for(int i = 1; i <= 5; i++) {
    for(int j = 1; j <= i; j++) {
        cout << ch;
    }
    ch++;
    cout << endl;
}
```

Pattern 16: Palindrome Number Triangle

```
    1
   121
  12321
 1234321
123454321
```

Code:

```
int n = 5;

for(int i = 1; i <= n; i++) {
    // Print spaces
    for(int j = 1; j <= n-i; j++) {
        cout << " ";
    }
    // Print increasing numbers
    for(int j = 1; j <= i; j++) {
        cout << j;
    }

    // Print decreasing numbers
    for(int j = i-1; j >= 1; j--) {
        cout << j;
    }

    cout << endl;
}
```

Pattern 17: Butterfly Pattern

```
*        *
**      **
***    ***
****  ****
**********
**********
****  ****
***    ***
**      **
*        *
```

Code:

```
int n = 5;

// Upper half
for(int i = 1; i <= n; i++) {
    // Left stars
    for(int j = 1; j <= i; j++) {
        cout << "*";
    }
    // Spaces
    for(int j = 1; j <= 2*(n-i); j++) {
        cout << " ";
    }
    // Right stars
    for(int j = 1; j <= i; j++) {
        cout << "*";
    }
    cout << endl;
}

// Lower half
for(int i = n; i >= 1; i--) {
    // Left stars
    for(int j = 1; j <= i; j++) {
        cout << "*";
    }
    // Spaces
    for(int j = 1; j <= 2*(n-i); j++) {
        cout << " ";
    }
    // Right stars
    for(int j = 1; j <= i; j++) {
        cout << "*";
    }
    cout << endl;
}
```

Pattern 18: Hollow Diamond

```
    *
   * *
  *   *
 *     *
*       *
 *     *
  *   *
   * *
    *
```

Code:

```
int n = 5;

// Upper half
for(int i = 1; i <= n; i++) {
    // Spaces before
    for(int j = 1; j <= n-i; j++) {
        cout << " ";
    }

    // Stars and spaces
    for(int j = 1; j <= 2*i-1; j++) {
        if(j == 1 || j == 2*i-1) {
            cout << "*";
        } else {
            cout << " ";
        }
    }
    cout << endl;
}

// Lower half
for(int i = n-1; i >= 1; i--) {
    for(int j = 1; j <= n-i; j++) {
        cout << " ";
    }
    for(int j = 1; j <= 2*i-1; j++) {
        if(j == 1 || j == 2*i-1) {
            cout << "*";
        } else {
            cout << " ";
        }
    }
    cout << endl;
}
```

Pattern 19: Zig-Zag Pattern

```
  *   *
 * * * *
*   *   *
```

Code:

```
int n = 9; // columns

for(int i = 1; i <= 3; i++) {
    for(int j = 1; j <= n; j++) {
        if((i + j) % 4 == 0 || (i == 2 && j % 4 == 0)) {
            cout << "*";
        } else {
            cout << " ";
        }
    }
    cout << endl;
}
```

Pattern 20: Plus Pattern

```
  *
  *
*****
  *
  *
```

Code:

```
int n = 5;
int mid = n/2 + 1;

for(int i = 1; i <= n; i++) {
    for(int j = 1; j <= n; j++) {
        if(i == mid || j == mid) {
            cout << "*";
        } else {
            cout << " ";
        }
    }
    cout << endl;
}
```

Part 2: Type Conversion

What is Type Conversion?

Converting one data type to another.

Two types:

1. **Implicit** (Automatic)
2. **Explicit** (Manual)

2.1 Implicit Type Conversion (Automatic)

Compiler automatically converts smaller type to larger type.

Example 1: int to double

```
int a = 10;
double b = a;  // Automatic conversion

cout << b << endl;  // 10.0
```

No data loss (10 becomes 10.0)

Example 2: char to int

```
char ch = 'A';
int num = ch;  // Automatic conversion

cout << num << endl;  // 65 (ASCII value)
```

Example 3: int to float

```
int x = 5;
float y = x;

cout << y << endl;  // 5.0
```

Type Hierarchy (Smaller → Larger)

```
char → short → int → long → float → double
```

Safe conversions:

- Smaller to larger ✅
- No data loss ✅

Dangerous Implicit Conversion

Example 1: double to int (Data Loss!)

```
double pi = 3.14159;
int num = pi;  // Automatic but LOSES decimal part

cout << num << endl;  // 3 (lost 0.14159!)
```

Example 2: Large to Small (Overflow!)

```
int big = 100000;
short small = big;  // May overflow

cout << small << endl;  // Unpredictable!
```

2.2 Explicit Type Conversion (Type Casting)

Manual conversion using cast operators.

Method 1: C-Style Cast

```
(type)variable
```

Example:

```
double pi = 3.14159;
int num = (int)pi;  // Explicit cast

cout << num << endl;  // 3
```

Method 2: C++ Style Cast

```
static_cast<type>(variable)
```

Example:

```
double pi = 3.14159;
int num = static_cast<int>(pi);

cout << num << endl;  // 3
```

Why Use Explicit Casting?

Example 1: Division Problem

```
int a = 10;
int b = 3;
```

```
cout << a / b << endl;  // 3 (integer division!)

// Fix: Cast to double
cout << (double)a / b << endl;  // 3.33333
```

Without cast:

```
10 / 3 → both int → result is int → 3
```

With cast:

```
(double)10 / 3 → 10.0 / 3 → result is double → 3.33333
```

Example 2: Percentage Calculation

```
int marks = 85;
int total = 100;

// Wrong
int percentage = (marks / total) * 100;
cout << percentage << endl;  // 0 (why?)

// Correct
double percentage = ((double)marks / total) * 100;
cout << percentage << endl;  // 85
```

Why wrong?

```
85 / 100 → 0 (integer division)
0 * 100 → 0
```

Example 3: Average Calculation

```
int a = 10, b = 20, c = 30;

// Wrong
int avg = (a + b + c) / 3;
cout << avg << endl;  // 20 (loses decimal)

// Correct
double avg = (double)(a + b + c) / 3;
cout << avg << endl;  // 20.0
```

Type Conversion in Expressions

Rule: Result type = Largest type in expression

```
int + int = int
int + double = double
float + double = double
char + int = int
```

Examples:

```
int a = 10;
double b = 3.5;
```

```cpp
auto result = a + b;  // result is double (10 + 3.5 = 13.5)

int x = 5;
int y = 2;
double z = x / y;  // z = 2.0 (NOT 2.5!)
                   // Because 5/2 happens first as int
```

Practice Problems

Problem 1: Temperature Converter

```cpp
#include <iostream>
using namespace std;

int main() {
    double celsius;

    cout << "Enter temperature in Celsius: ";
    cin >> celsius;

    double fahrenheit = (celsius * 9.0 / 5.0) + 32;

    cout << celsius << "°C = " << fahrenheit << "°F" << endl;

    return 0;
}
```

Problem 2: Grade Percentage

```cpp
#include <iostream>
using namespace std;
```

```
int main() {
    int subject1, subject2, subject3;

    cout << "Enter marks of 3 subjects: ";
    cin >> subject1 >> subject2 >> subject3;

    int total = subject1 + subject2 + subject3;
    double percentage = (double)total / 3;

    cout << "Total: " << total << endl;
    cout << "Percentage: " << percentage << "%" << endl;

    return 0;
}
```

Problem 3: ASCII Value

```
#include <iostream>
using namespace std;

int main() {
    char ch;

    cout << "Enter a character: ";
    cin >> ch;

    int ascii = (int)ch;

    cout << "ASCII value of " << ch << " is " << ascii << endl;

    return 0;
}
```

Summary

Pattern Printing Tips:

1. Outer loop = rows
2. Inner loop = columns
3. Use spaces for alignment
4. Practice drawing on paper first

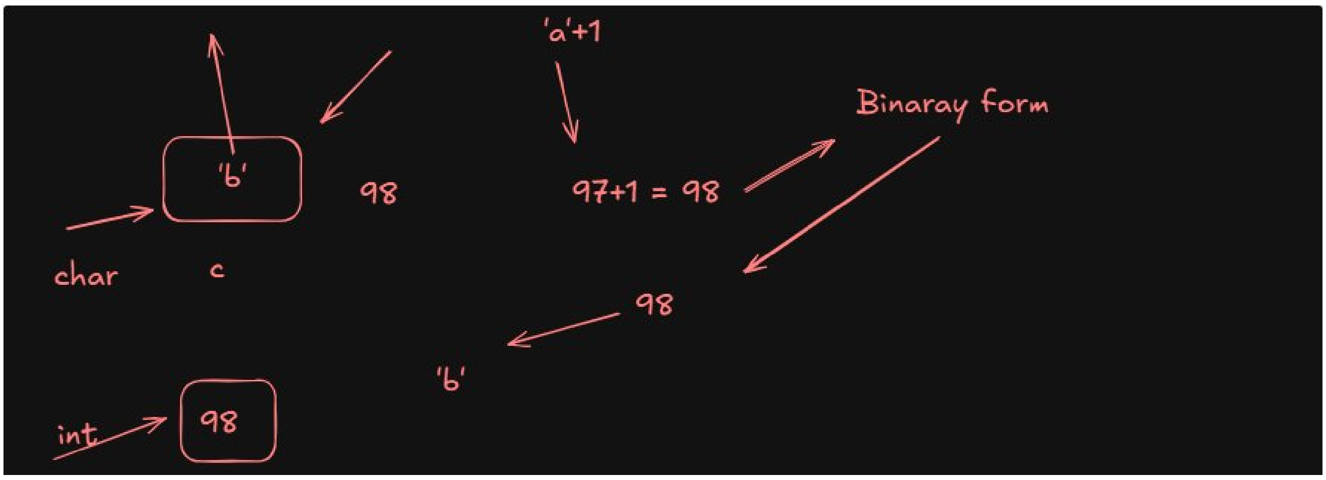
Type Conversion:

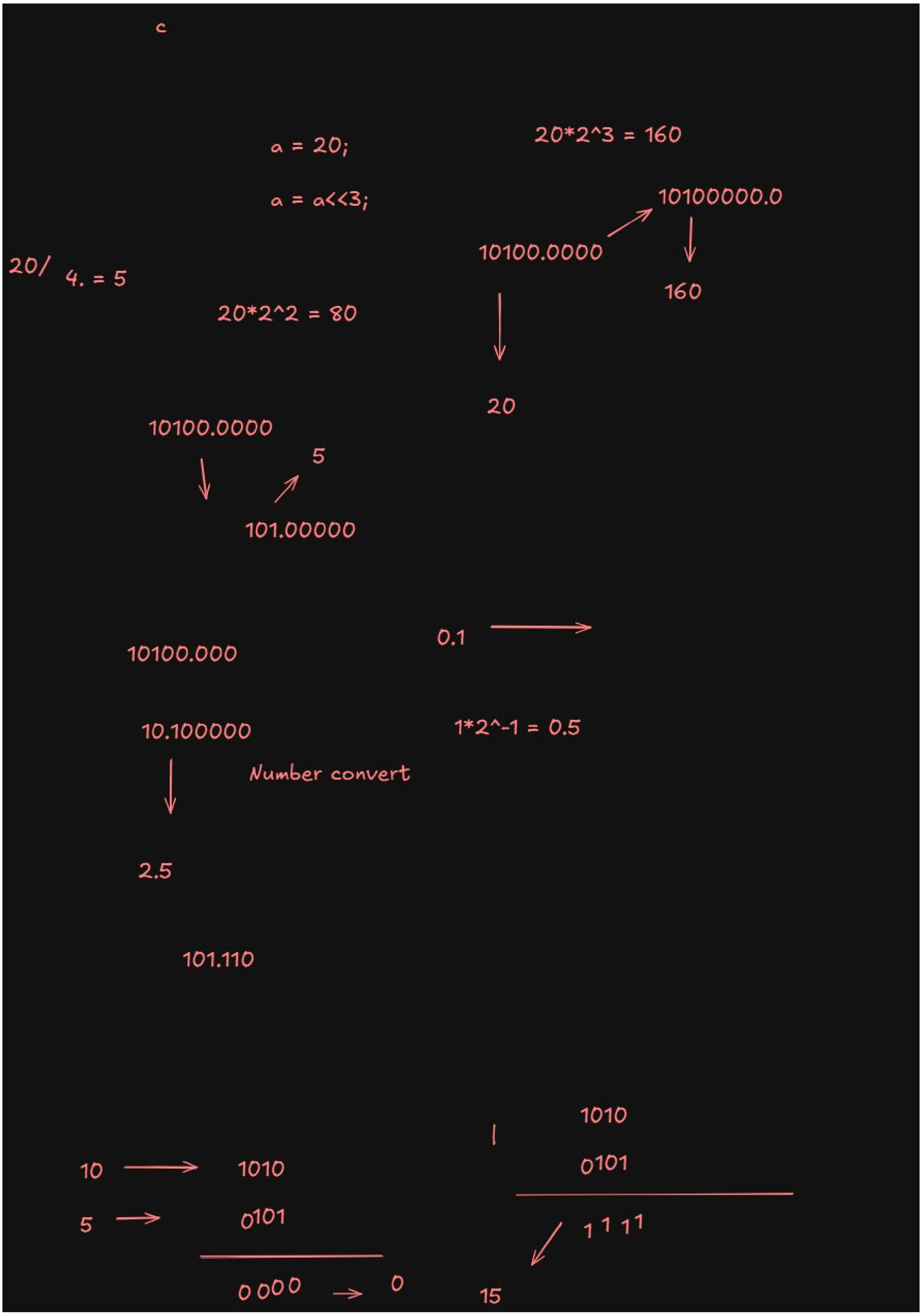
Type	When	Example
Implicit	Automatic	`int a = 5; double b = a;`
Explicit	Manual cast	`int x = (int)3.14;`

Common Mistakes:

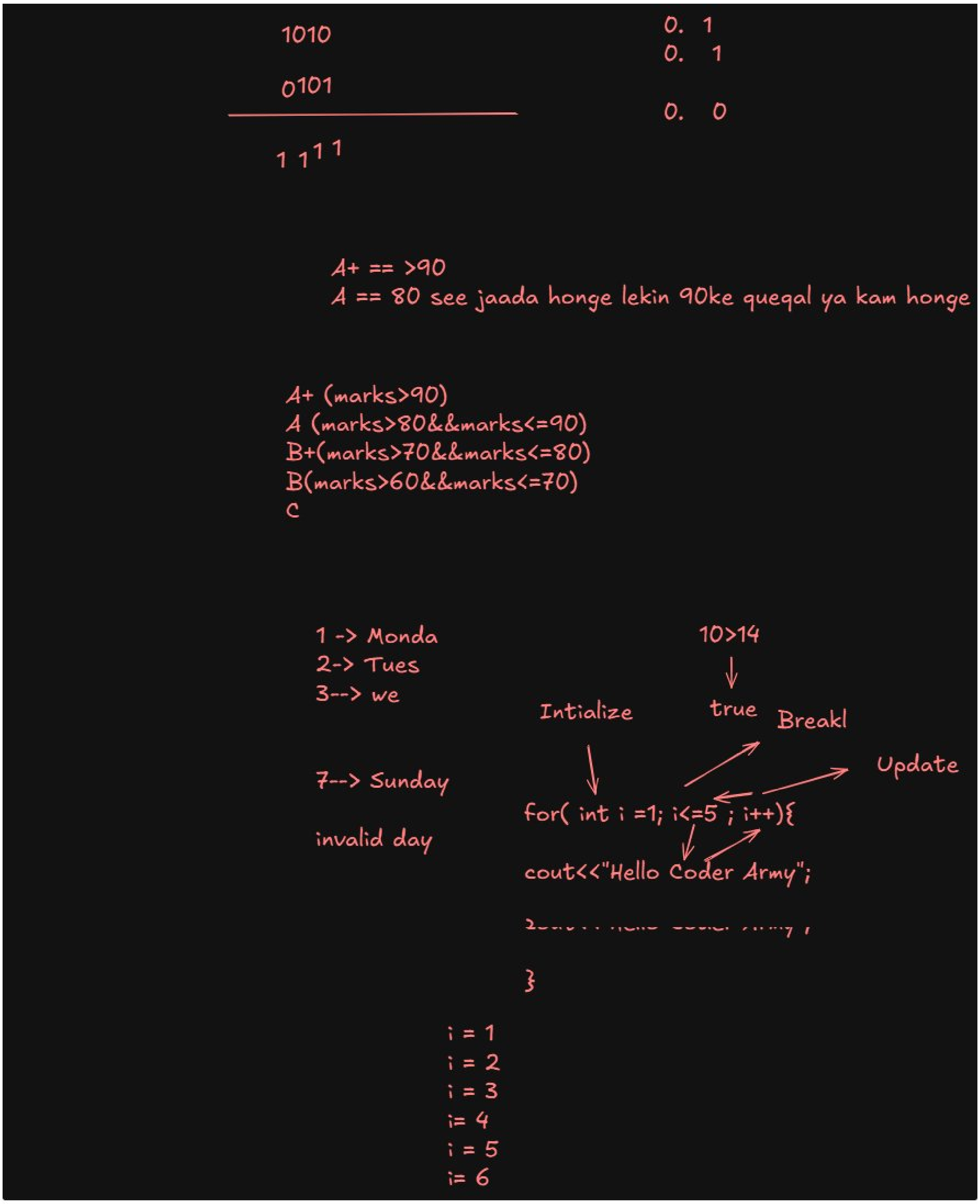
```
// ❌ Wrong
int result = 10 / 3;  // 3

// ✅ Correct
double result = (double)10 / 3;  // 3.33333
```




```
a = 20;

a = a<<3;
```
c
20*2^3 = 160
10100000.0
10100.0000
160
20/ 4. = 5
20*2^2 = 80
20
10100.0000
5
101.00000
0.1
10100.000
1*2^-1 = 0.5
10.100000
Number convert
2.5
101.110
1010
|
10
1010
0101
5
0101
1 1 1 1
0 000
0
15



```
1010

0101
________________

1 1 1 1
```

```
0.  1
0.  1

0.  0
```

A+ == >90
A == 80 see jaada honge lekin 90ke queqal ya kam honge

```
A+ (marks>90)
A (marks>80&&marks<=90)
B+(marks>70&&marks<=80)
B(marks>60&&marks<=70)
C
```

```
1 -> Monda
2-> Tues
3--> we


7--> Sunday

invalid day
```

```
for( int i =1; i<=5 ; i++){

cout<<"Hello Coder Army";

[illegible]

}
```

```
i = 1
i = 2
i = 3
i= 4
i = 5
i= 6
```