C Programming: Conditional Statements

Prof. Jyotiprakash Mishra
mail@jyotiprakash.org

January 16, 2026

Topics Covered

What are Conditional Statements?

- Control flow statements that execute code based on conditions
- Allow programs to make decisions
- Evaluate boolean expressions (true/false)
- In C: 0 is false, any non-zero value is true

Types of Conditional Statements:

1. `if` statement
2. `if-else` statement
3. `else-if` ladder
4. Nested conditionals
5. Ternary operator (`? :`)

if Statement - Syntax

Syntax:

```c
if (condition) {
  // code to execute if condition is true
}
```

Flow:

- Condition is evaluated
- If true (non-zero), execute the block
- If false (0), skip the block
- Continue with next statement

Program 1: Simple if Statement

```c
#include <stdio.h>
int main() {
  int age = 20;
  printf("Age: %d\n", age);
  if (age >= 18) {
    printf("You are an adult\n");
  }
  printf("Program continues\n");
  return 0;
}
```

Output:

```
Age: 20
You are an adult
Program continues
```

Explanation:

- age >= 18 is true
- Block executes
- Program continues after if

Program 2: if Statement - Condition False

```c
#include <stdio.h>
int main() {
  int age = 15;
  printf("Age: %d\n", age);
  if (age >= 18) {
    printf("You are an adult\n");
  }
  printf("Program continues\n");
  return 0;
}
```

Output:

```
Age: 15
Program continues
```

Explanation:

- age >= 18 is false
- Block is skipped
- Program continues

Program 3: Multiple if Statements

```c
#include <stdio.h>
int main() {
  int num = 10;
  printf("Number: %d\n\n", num);
  if (num > 0) {
    printf("Number is positive\n");
  }
  if (num % 2 == 0) {
    printf("Number is even\n");
  }
  if (num >= 10) {
    printf("Number is >= 10\n");
  }
  return 0;
}
```

Output:

```
Number: 10

Number is positive
Number is even
Number is >= 10
```

Note:

- All conditions checked
- Each if is independent
- All three execute

if-else Statement - Syntax

Syntax:

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

Flow:

- Condition is evaluated
- If true: execute if block, skip else block
- If false: skip if block, execute else block
- Exactly one block always executes

Program 4: if-else Statement

```c
#include <stdio.h>
int main() {
  int num = 7;
  printf("Number: %d\n", num);
  if (num % 2 == 0) {
    printf("Number is even\n");
  } else {
    printf("Number is odd\n");
  }
  return 0;
}
```

Output:

```
Number: 7
Number is odd
```

Explanation:

- 7 % 2 == 0 is false
- if block skipped
- else block executes

Program 5: if-else - Positive/Negative

```c
#include <stdio.h>
int main() {
  int num = -5;
  printf("Number: %d\n", num);
  if (num >= 0) {
    printf("Number is non-negative\n");
  } else {
    printf("Number is negative\n");
  }
  return 0;
}
```

Output:

```
Number: -5
Number is negative
```

Explanation:

- $-5 >= 0$ is false
- else block executes

Program 6: if-else - Voting Eligibility

```c
#include <stdio.h>
int main() {
  int age = 17;
  printf("Age: %d\n\n", age);
  if (age >= 18) {
    printf("Eligible to vote\n");
    printf("Please register!\n");
  } else {
    printf("Not eligible to vote\n");
    printf("Wait %d year(s)\n",
           18 - age);
  }
  return 0;
}
```

Output:

```
Age: 17

Not eligible to vote
Wait 1 year(s)
```

Note:

- Multiple statements in else
- Calculated waiting years

else-if Ladder - Syntax

Syntax:

```
if (condition1) {
  // code if condition1 is true
} else if (condition2) {
  // code if condition2 is true
} else if (condition3) {
  // code if condition3 is true
} else {
  // code if all conditions are false
}
```

Flow:

- Conditions checked top to bottom
- First true condition's block executes
- Remaining conditions skipped
- else block executes if all false

Program 7: else-if Ladder - Grades

```c
#include <stdio.h>
int main() {
  int marks = 75;
  printf("Marks: %d\n", marks);
  if (marks >= 90) {
    printf("Grade: A+\n");
  } else if (marks >= 80) {
    printf("Grade: A\n");
  } else if (marks >= 70) {
    printf("Grade: B\n");
  } else if (marks >= 60) {
    printf("Grade: C\n");
  } else {
    printf("Grade: F\n");
  }
  return 0;
}
```

Output:

```
Marks: 75
Grade: B
```

Explanation:

- `marks >= 90`: false
- `marks >= 80`: false
- `marks >= 70`: true
- Grade B printed, rest skipped

Program 8: else-if - Number Categories

```c
#include <stdio.h>
int main() {
  int num = 0;
  printf("Number: %d\n", num);
  if (num > 0) {
    printf("Positive number\n");
  } else if (num < 0) {
    printf("Negative number\n");
  } else {
    printf("Zero\n");
  }
  return 0;
}
```

Output:

```
Number: 0
Zero
```

Explanation:

- num > 0: false
- num < 0: false
- else block executes

Program 9: else-if - Age Categories

```c
#include <stdio.h>
int main() {
  int age = 35;
  printf("Age: %d\n", age);
  if (age < 0) {
    printf("Invalid age\n");
  } else if (age < 13) {
    printf("Child\n");
  } else if (age < 20) {
    printf("Teenager\n");
  } else if (age < 60) {
    printf("Adult\n");
  } else {
    printf("Senior citizen\n");
  }
  return 0;
}
```

Output:

```
Age: 35
Adult
```

Note:

- Multiple age ranges
- First match wins
- Order matters!

Program 10: else-if - Days of Week

```c
#include <stdio.h>
int main() {
  int day = 3;
  printf("Day number: %d\n", day);
  if (day == 1) {
    printf("Monday\n");
  } else if (day == 2) {
    printf("Tuesday\n");
  } else if (day == 3) {
    printf("Wednesday\n");
  } else if (day == 4) {
    printf("Thursday\n");
  } else if (day == 5) {
    printf("Friday\n");
  } else if (day == 6) {
    printf("Saturday\n");
  } else if (day == 7) {
    printf("Sunday\n");
  } else {
    printf("Invalid day\n");
  }
  return 0;
}
```

Output:

```
Day number: 3
Wednesday
```

Note:

- Exact value matching
- Handles invalid input
- Better with switch-case

Nested Conditionals

Definition:

- if/else statements inside other if/else statements
- Creates decision trees
- Can have multiple levels of nesting

Syntax:

```
if (condition1) {
  if (condition2) {
    // executes if both true
  } else {
    // executes if condition1 true, condition2 false
  }
} else {
  // executes if condition1 false
}
```

Program 11: Nested if - Number Check

```c
#include <stdio.h>
int main() {
  int num = 12;
  printf("Number: %d\n\n", num);
  if (num > 0) {
    printf("Positive number\n");
    if (num % 2 == 0) {
      printf("Even number\n");
    } else {
      printf("Odd number\n");
    }
  } else {
    printf("Not a positive number\n");
  }
  return 0;
}
```

Output:

```
Number: 12

Positive number
Even number
```

Logic:

- Outer: checks if positive
- Inner: checks if even/odd
- Inner only runs if outer true

Program 12: Nested if - Login System

```c
#include <stdio.h>
int main() {
  int user_exists = 1;
  int password_correct = 1;
  printf("User exists: %d\n",
         user_exists);
  printf("Password correct: %d\n\n",
         password_correct);
  if (user_exists) {
    if (password_correct) {
      printf("Login successful\n");
    } else {
      printf("Wrong password\n");
    }
  } else {
    printf("User not found\n");
  }
  return 0;
}
```

Output:

```
User exists: 1
Password correct: 1

Login successful
```

Logic:

- First check: user exists
- Then check: password
- Two-level authentication

Program 13: Nested if-else - Divisibility

```c
#include <stdio.h>
int main() {
  int num = 15;
  printf("Number: %d\n\n", num);
  if (num % 3 == 0) {
    if (num % 5 == 0) {
      printf("Divisible by both 3 and 5\n");
      printf("FizzBuzz!\n");
    } else {
      printf("Divisible by 3 only\n");
      printf("Fizz!\n");
    }
  } else if (num % 5 == 0) {
    printf("Divisible by 5 only\n");
    printf("Buzz!\n");
  } else {
    printf("Not divisible by 3 or 5\n");
  }
  return 0;
}
```

Output:

```
Number: 15

Divisible by both 3 and 5
FizzBuzz!
```

Note:

- Classic FizzBuzz logic
- Nested and chained
- Checks all combinations

Program 14: Deep Nesting - Triangle Validity

```c
#include <stdio.h>
int main() {
  int a = 3, b = 4, c = 5;
  printf("Sides: %d, %d, %d\n\n",
         a, b, c);
  if (a > 0 && b > 0 && c > 0) {
    if (a + b > c) {
      if (b + c > a) {
        if (c + a > b) {
          printf("Valid triangle\n");
        } else {
          printf("Invalid: c+a <= b\n");
        }
      } else {
        printf("Invalid: b+c <= a\n");
      }
    } else {
      printf("Invalid: a+b <= c\n");
    }
  } else {
    printf("Invalid: negative side\n");
  }
  return 0;
}
```

Output:

```
Sides: 3, 4, 5

Valid triangle
```

Note:

- 4 levels deep
- Triangle inequality
- Each check necessary

Ternary Operator (? :)

Syntax:

```
condition ? expression_if_true : expression_if_false
```

Features:

- Shorthand for simple if-else
- Returns a value
- Can be used in expressions and assignments
- More concise for simple conditions

Equivalent to:

```c
if (condition) {
  result = expression_if_true;
} else {
  result = expression_if_false;
}
```

Program 15: Basic Ternary Operator

```c
#include <stdio.h>
int main() {
  int a = 10, b = 20;
  int max;
  printf("a = %d, b = %d\n\n", a, b);
  max = (a > b) ? a : b;
  printf("Using ternary:\n");
  printf("Maximum = %d\n\n", max);
  if (a > b) {
    max = a;
  } else {
    max = b;
  }
  printf("Using if-else:\n");
  printf("Maximum = %d\n", max);
  return 0;
}
```

Output:

```
a = 10, b = 20

Using ternary:
Maximum = 20

Using if-else:
Maximum = 20
```

Explanation:

- `a > b` is false
- Returns `b` (20)
- Same as if-else

Program 16: Ternary in printf

```c
#include <stdio.h>
int main() {
  int num = 7;
  printf("Number: %d\n", num);
  printf("The number is %s\n",
         (num % 2 == 0) ? "even" : "odd");
  int age = 25;
  printf("\nAge: %d\n", age);
  printf("You are %s\n",
         (age >= 18) ? "an adult" :
                       "a minor");
  return 0;
}
```

Output:

```
Number: 7
The number is odd

Age: 25
You are an adult
```

Note:

- Used directly in printf
- Returns string literals
- Very concise

Program 17: Nested Ternary Operators

```c
#include <stdio.h>
int main() {
  int num = 0;
  char *result;
  printf("Number: %d\n", num);
  result = (num > 0) ? "Positive" :
           (num < 0) ? "Negative" :
                       "Zero";
  printf("Result: %s\n", result);
  int a = 15, b = 20, c = 10;
  int max;
  printf("\na = %d, b = %d, c = %d\n",
         a, b, c);
  max = (a > b) ?
        ((a > c) ? a : c) :
        ((b > c) ? b : c);
  printf("Maximum: %d\n", max);
  return 0;
}
```

Output:

```
Number: 0
Result: Zero

a = 15, b = 20, c = 10
Maximum: 20
```

Warning:

- Nested ternary works
- Can be hard to read
- Use with caution

Program 18: Ternary for Absolute Value

```c
#include <stdio.h>
int main() {
  int num = -15;
  int absolute;
  printf("Number: %d\n", num);
  absolute = (num < 0) ? -num : num;
  printf("Absolute value: %d\n\n",
         absolute);
  num = 25;
  printf("Number: %d\n", num);
  absolute = (num < 0) ? -num : num;
  printf("Absolute value: %d\n",
         absolute);
  return 0;
}
```

Output:

```
Number: -15
Absolute value: 15

Number: 25
Absolute value: 25
```

Logic:

- If negative, negate it
- If positive, keep it
- Compact absolute value

Program 19: Common Mistake - Assignment vs Comparison

```c
#include <stdio.h>
int main() {
  int x = 5;
  printf("x = %d\n\n", x);
  printf("Wrong (assignment):\n");
  if (x = 10) {
    printf("  This always executes!\n");
    printf("  x is now %d\n\n", x);
  }
  x = 5;
  printf("Correct (comparison):\n");
  if (x == 10) {
    printf("  This won't execute\n");
  } else {
    printf("  x is still %d\n", x);
  }
  return 0;
}
```

Output:

```
x = 5

Wrong (assignment):
  This always executes!
  x is now 10

Correct (comparison):
  x is still 5
```

Warning:

- = assigns, returns value
- == compares
- Always use ==!

Program 20: Logical AND in Conditions

```c
#include <stdio.h>
int main() {
  int age = 25;
  int has_license = 1;
  printf("Age: %d\n", age);
  printf("Has license: %d\n\n",
         has_license);
  if (age >= 18 && has_license) {
    printf("Can drive legally\n");
  } else {
    printf("Cannot drive:\n");
    if (age < 18) {
      printf("  Too young\n");
    }
    if (!has_license) {
      printf("  No license\n");
    }
  }
  return 0;
}
```

Output:

```
Age: 25
Has license: 1

Can drive legally
```

Note:

- Both conditions must be true
- && for logical AND
- Short-circuit evaluation

Program 21: Logical OR in Conditions

```c
#include <stdio.h>
int main() {
  char grade = 'B';
  printf("Grade: %c\n\n", grade);
  if (grade == 'A' || grade == 'B') {
    printf("Excellent performance!\n");
    printf("Scholarship eligible\n");
  } else if (grade == 'C') {
    printf("Good performance\n");
  } else {
    printf("Need improvement\n");
  }
  return 0;
}
```

Output:

```
Grade: B

Excellent performance!
Scholarship eligible
```

Note:

- At least one must be true
- `||` for logical OR
- Short-circuit evaluation

Program 22: Logical NOT in Conditions

```c
#include <stdio.h>
int main() {
  int is_raining = 0;
  int is_sunny = 1;
  printf("Raining: %d, Sunny: %d\n\n",
         is_raining, is_sunny);
  if (!is_raining) {
    printf("No umbrella needed\n");
  }
  if (!is_sunny) {
    printf("Take sunscreen\n");
  } else {
    printf("Sun protection advised\n");
  }
  return 0;
}
```

Output:

```
Raining: 0, Sunny: 1

No umbrella needed
Sun protection advised
```

Note:

- ! negates condition
- !0 is true (1)
- !1 is false (0)

Conditional Statements - Summary

Types:

- **if**: Execute block if condition is true
- **if-else**: Choose between two alternatives
- **else-if ladder**: Multiple mutually exclusive conditions
- **Nested conditionals**: Conditions inside conditions
- **Ternary operator**: Compact conditional expression

Key Points:

- Always use == for comparison, not =
- Use && for AND, || for OR, ! for NOT
- In C: 0 is false, non-zero is true
- Ternary is concise but can hurt readability when nested
- Order matters in else-if ladder
- Avoid deep nesting when possible

Best Practices

1. **Use braces** even for single statements
2. **Keep conditions simple** and readable
3. **Avoid deep nesting** - refactor if >3 levels
4. **Check edge cases** (zero, negative, boundary values)
5. **Use ternary sparingly** - only for simple cases
6. **Order else-if conditions** from most to least specific
7. **Validate input** before making decisions
8. **Use descriptive variable names** for boolean conditions

Practice Exercises

Try these programs:

1. Check if a year is a leap year
2. Find the largest of three numbers
3. Calculate tax based on income slabs
4. Determine triangle type (equilateral, isosceles, scalene)
5. Convert numeric grade to letter grade
6. Check if a character is vowel or consonant
7. Implement a simple calculator with $+$, -, *, $/$
8. Check password strength (length, digits, special chars)