C Programming: Switch-Case

Prof. Jyotiprakash Mishra
mail@jyotiprakash.org

January 16, 2026

Topics Covered

What is Switch-Case?

- Multi-way decision statement
- Alternative to multiple if-else statements
- Tests a variable against multiple values
- More readable for discrete value checks
- Can only test for equality (not ranges)

When to Use:

- Multiple specific values to test
- Integer or character values
- Menu-driven programs
- State machines

Switch-Case Syntax

Syntax:

```c
switch (expression) {
  case constant1:
    // code for constant1
    break;
  case constant2:
    // code for constant2
    break;
  default:
    // code if no case matches
}
```

Key Points:

- Expression must evaluate to int or char
- Cases must be compile-time constants
- break exits the switch
- default is optional but recommended

Program 1: Simple Switch-Case

```c
#include <stdio.h>
int main() {
  int day = 3;
  printf("Day number: %d\n", day);
  printf("Day name: ");
  switch (day) {
    case 1:
      printf("Monday\n");
      break;
    case 2:
      printf("Tuesday\n");
      break;
    case 3:
      printf("Wednesday\n");
      break;
    case 4:
      printf("Thursday\n");
      break;
    default:
      printf("Invalid\n");
  }
  return 0;
}
```

Output:

```
Day number: 3
Day name: Wednesday
```

Explanation:

- Expression: day (value 3)
- Matches case 3
- Prints "Wednesday"
- break exits switch

Program 2: Switch with Default

```
#include <stdio.h>
int main() {
  int num = 10;
  printf("Number: %d\n", num);
  printf("Category: ");
  switch (num) {
    case 1:
      printf("One\n");
      break;
    case 2:
      printf("Two\n");
      break;
    case 3:
      printf("Three\n");
      break;
    default:
      printf("Other number\n");
  }
  return 0;
}
```

Output:

```
Number: 10
Category: Other number
```

Explanation:

- No case matches 10
- default executes
- Handles unexpected values
- Good practice to include

Program 3: Character Switch

```c
#include <stdio.h>
int main() {
  char grade = 'B';
  printf("Grade: %c\n", grade);
  printf("Performance: ");
  switch (grade) {
    case 'A':
      printf("Excellent!\n");
      break;
    case 'B':
      printf("Good job!\n");
      break;
    case 'C':
      printf("Fair\n");
      break;
    case 'D':
      printf("Poor\n");
      break;
    default:
      printf("Invalid grade\n");
  }
  return 0;
}
```

Output:

```
Grade: B
Performance: Good job!
```

Note:

- Switch works with char
- Character in single quotes
- Case-sensitive: 'A' != 'a'

Multiple Cases - Same Action

Syntax:

```c
switch (value) {
  case 1:
  case 2:
  case 3:
    // code for 1, 2, or 3
    break;
  case 4:
  case 5:
    // code for 4 or 5
    break;
}
```

Purpose:

- Multiple values trigger same code
- No break between grouped cases
- Falls through until break

Program 4: Vowel or Consonant

```c
#include <stdio.h>
int main() {
  char ch = 'e';
  printf("Character: %c\n", ch);
  switch (ch) {
    case 'a':
    case 'e':
    case 'i':
    case 'o':
    case 'u':
      printf("Vowel\n");
      break;
    default:
      printf("Consonant\n");
  }
  return 0;
}
```

Output:

```
Character: e
Vowel
```

Explanation:

- 5 cases for vowels
- No break between them
- Falls through to printf
- All vowels execute same code

Program 5: Weekday or Weekend

```c
#include <stdio.h>
int main() {
  int day = 6;
  printf("Day number: %d\n", day);
  switch (day) {
    case 1:
    case 2:
    case 3:
    case 4:
    case 5:
      printf("Weekday\n");
      printf("Go to work!\n");
      break;
    case 6:
    case 7:
      printf("Weekend\n");
      printf("Relax!\n");
      break;
    default:
      printf("Invalid day\n");
  }
  return 0;
}
```

Output:

```
Day number: 6
Weekend
Relax!
```

Note:

- Cases 1-5: Weekday
- Cases 6-7: Weekend
- Grouping related values

Program 6: Upper and Lower Case

```c
#include <stdio.h>
int main() {
  char ch = 'A';
  printf("Character: %c\n", ch);
  switch (ch) {
    case 'a':
    case 'A':
      printf("Letter A\n");
      break;
    case 'b':
    case 'B':
      printf("Letter B\n");
      break;
    case 'c':
    case 'C':
      printf("Letter C\n");
      break;
    default:
      printf("Other letter\n");
  }
  return 0;
}
```

Output:

```
Character: A
Letter A
```

Note:

- Handles both cases
- 'A' and 'a' same result
- Case-insensitive matching

Fall-Through in Switch

What is Fall-Through?

- Execution continues to next case
- Happens when break is missing
- Can be intentional or a bug
- All subsequent cases execute

Intentional Fall-Through:

- Grouping multiple cases
- Cascading actions

Accidental Fall-Through:

- Common mistake: forgetting break
- Causes unexpected behavior
- Hard to debug

Program 7: Fall-Through Demo

```c
#include <stdio.h>
int main() {
  int num = 2;
  printf("Number: %d\n", num);
  printf("Output:\n");
  switch (num) {
    case 1:
      printf("Case 1\n");
    case 2:
      printf("Case 2\n");
    case 3:
      printf("Case 3\n");
    default:
      printf("Default\n");
  }
  return 0;
}
```

Output:

```
Number: 2
Output:
Case 2
Case 3
Default
```

Warning:

- No break statements!
- Starts at case 2
- Falls through case 3
- Executes default too
- Usually a bug

Program 8: Fall-Through Fixed

```c
#include <stdio.h>
int main() {
  int num = 2;
  printf("Number: %d\n", num);
  printf("Output:\n");
  switch (num) {
    case 1:
      printf("Case 1\n");
      break;
    case 2:
      printf("Case 2\n");
      break;
    case 3:
      printf("Case 3\n");
      break;
    default:
      printf("Default\n");
  }
  return 0;
}
```

Output:

```
Number: 2
Output:
Case 2
```

Correct:

- break after each case
- Only case 2 executes
- Exits switch properly
- Expected behavior

Program 9: Intentional Fall-Through

```
#include <stdio.h>
int main() {
  int month = 3;
  int days;
  printf("Month: %d\n", month);
  switch (month) {
    case 1: case 3: case 5:
    case 7: case 8: case 10:
    case 12:
      days = 31;
      break;
    case 4: case 6:
    case 9: case 11:
      days = 30;
      break;
    case 2:
      days = 28;
      break;
    default:
      days = 0;
  }
  printf("Days: %d\n", days);
  return 0;
}
```

Output:

```
Month: 3
Days: 31
```

Good Use:

- Intentional fall-through
- Groups months by days
- Clear and concise
- Better than if-else

Program 10: Simple Calculator

```c
#include <stdio.h>
int main() {
  int a = 10, b = 5;
  char op = '*';
  int result;
  printf("Expression: %d %c %d\n",
         a, op, b);
  switch (op) {
    case '+':
      result = a + b;
      break;
    case '-':
      result = a - b;
      break;
    case '*':
      result = a * b;
      break;
    case '/':
      result = a / b;
      break;
    default:
      printf("Invalid op\n");
      return 1;
  }
  printf("Result: %d\n", result);
  return 0;
}
```

Output:

```
Expression: 10 * 5
Result: 50
```

Note:

- Character operator
- Four operations
- Clean and readable

Program 11: Calculator with Validation

```c
#include <stdio.h>
int main() {
  int a = 10, b = 0;
  char op = '/';
  printf("%d %c %d = ", a, op, b);
  switch (op) {
    case '+':
      printf("%d\n", a + b);
      break;
    case '-':
      printf("%d\n", a - b);
      break;
    case '*':
      printf("%d\n", a * b);
      break;
    case '/':
      if (b == 0) {
        printf("Error: Div by 0\n");
      } else {
        printf("%d\n", a / b);
      }
      break;
    default:
      printf("Invalid op\n");
  }
  return 0;
}
```

Output:

```
10 / 0 = Error: Div by 0
```

Note:

- Checks division by zero
- Validation inside case
- Prevents crash

Program 12: Simple Menu

```c
#include <stdio.h>
int main() {
  int choice = 2;
  printf("Menu:\n");
  printf("1. Start\n");
  printf("2. Stop\n");
  printf("3. Pause\n");
  printf("Choice: %d\n\n", choice);
  switch (choice) {
    case 1:
      printf("Starting...\n");
      break;
    case 2:
      printf("Stopping...\n");
      break;
    case 3:
      printf("Pausing...\n");
      break;
    default:
      printf("Invalid choice\n");
  }
  return 0;
}
```

Output:

```
Menu:
1. Start
2. Stop
3. Pause
Choice: 2

Stopping...
```

Note:

- Common menu pattern
- Integer choices
- Default for invalid input

Program 13: Multi-Level Menu

```c
#include <stdio.h>
int main() {
  int main_choice = 1;
  int sub_choice = 2;
  printf("Main: %d, Sub: %d\n\n",
         main_choice, sub_choice);
  switch (main_choice) {
    case 1:
      printf("File menu:\n");
      switch (sub_choice) {
        case 1:
          printf("  New\n");
          break;
        case 2:
          printf("  Open\n");
          break;
        case 3:
          printf("  Save\n");
          break;
      }
      break;
    case 2:
      printf("Edit menu\n");
      break;
  }
  return 0;
}
```

Output:

```
Main: 1, Sub: 2

File menu:
  Open
```

Note:

- Nested switch statements
- Two-level menu
- Each switch independent

Program 14: Switch with Expression

```c
#include <stdio.h>
int main() {
  int num = 17;
  printf("Number: %d\n", num);
  printf("Remainder when div by 5:\n");
  switch (num % 5) {
    case 0:
      printf("Divisible by 5\n");
      break;
    case 1:
      printf("Remainder 1\n");
      break;
    case 2:
      printf("Remainder 2\n");
      break;
    case 3:
      printf("Remainder 3\n");
      break;
    case 4:
      printf("Remainder 4\n");
      break;
  }
  return 0;
}
```

Output:

```
Number: 17
Remainder when div by 5:
Remainder 2
```

Note:

- Expression: `num % 5`
- Evaluated once
- Result matched to cases
- `17 % 5 = 2`

Program 15: Switch with Function Call

```c
#include <stdio.h>
#include <ctype.h>
int main() {
  char ch = 'A';
  printf("Character: %c\n", ch);
  switch (tolower(ch)) {
    case 'a':
      printf("Letter A (any case)\n");
      break;
    case 'b':
      printf("Letter B (any case)\n");
      break;
    case 'c':
      printf("Letter C (any case)\n");
      break;
    default:
      printf("Other letter\n");
  }
  return 0;
}
```

Output:

```
Character: A
Letter A (any case)
```

Note:

- tolower() converts to lowercase
- Function called once
- Only lowercase cases needed
- Cleaner than duplicate cases

Program 16: Number Classification

```c
#include <stdio.h>
int main() {
  int num = 0;
  printf("Number: %d\n", num);
  switch (num) {
    case 0:
      printf("Zero\n");
      break;
    case 1:
      printf("One (unit)\n");
      break;
    case 2:
    case 3:
    case 5:
    case 7:
      printf("Small prime\n");
      break;
    case 4:
    case 6:
    case 8:
    case 9:
      printf("Small composite\n");
      break;
    default:
      printf("Larger number\n");
  }
  return 0;
}
```

Output:

```
Number: 0
Zero
```

Note:

- Groups related numbers
- Primes vs composites
- Special case for 0 and 1

Program 17: ASCII Code Checker

```c
#include <stdio.h>
int main() {
  char ch = '5';
  printf("Character: '%c'\n", ch);
  printf("Type: ");
  switch (ch) {
    case '0': case '1': case '2':
    case '3': case '4': case '5':
    case '6': case '7': case '8':
    case '9':
      printf("Digit\n");
      break;
    case '+': case '-':
    case '*': case '/':
      printf("Operator\n");
      break;
    case ' ':
      printf("Space\n");
      break;
    default:
      printf("Other\n");
  }
  return 0;
}
```

Output:

```
Character: '5'
Type: Digit
```

Note:

- Character classification
- Digits '0' to '9'
- Operators grouped
- Space as special case

Switch vs If-Else Comparison

Switch-Case	**If-Else**
Tests equality only	Can test any condition
Integer/char values	Any boolean expression
Constant values only	Variables, ranges, expressions
More readable for many values	Better for few conditions
Can be optimized by compiler	Sequential checking
Fall-through possible	No fall-through

Use Switch When:

- Testing one variable against many constant values
- Values are discrete integers or characters
- Menu systems, state machines

Program 18: Switch vs If-Else - Same Logic

```c
#include <stdio.h>
int main() {
  int grade = 85;
  printf("Grade: %d\n\n", grade);
  printf("Using if-else:\n");
  if (grade >= 90) {
    printf("A\n");
  } else if (grade >= 80) {
    printf("B\n");
  } else if (grade >= 70) {
    printf("C\n");
  } else {
    printf("F\n");
  }
  printf("\nNote: Switch can't\n");
  printf("do ranges easily\n");
  return 0;
}
```

Output:

```
Grade: 85

Using if-else:
B

Note: Switch can't
do ranges easily
```

Note:

- Range checking needs if-else
- Switch only for exact values
- if-else more flexible here

Program 19: When Switch is Better

```c
#include <stdio.h>
int main() {
  int code = 404;
  printf("HTTP Code: %d\n", code);
  printf("Message: ");
  switch (code) {
    case 200:
      printf("OK\n");
      break;
    case 404:
      printf("Not Found\n");
      break;
    case 500:
      printf("Server Error\n");
      break;
    case 403:
      printf("Forbidden\n");
      break;
    default:
      printf("Unknown\n");
  }
  return 0;
}
```

Output:

```
HTTP Code: 404
Message: Not Found
```

Why Switch Better:

- Many discrete values
- More readable
- Clear intent
- Easier to maintain

Program 20: Mistake - Missing Break

```c
#include <stdio.h>
int main() {
  int level = 2;
  printf("Level: %d\n", level);
  printf("Access:\n");
  switch (level) {
    case 1:
      printf("  Basic\n");
    case 2:
      printf("  Intermediate\n");
    case 3:
      printf("  Advanced\n");
    default:
      printf("  Unknown\n");
  }
  printf("\nBUG: Missing breaks!\n");
  return 0;
}
```

Output:

```
Level: 2
Access:
  Intermediate
  Advanced
  Unknown

BUG: Missing breaks!
```

Problem:

- Forgot break statements
- Falls through all cases
- Unintended behavior
- Common beginner mistake

Program 21: Mistake - Variable in Case

```c
#include <stdio.h>
int main() {
  int x = 5;
  int y = 5;
  printf("This won't compile:\n\n");
  printf("switch (x) {\n");
  printf("  case y:  // ERROR!\n");
  printf("    ...\n");
  printf("}\n\n");
  printf("Case must be constant,\n");
  printf("not variable!\n");
  return 0;
}
```

Output:

```
This won't compile:

switch (x) {
  case y:  // ERROR!
    ...
}

Case must be constant,
not variable!
```

Rule:

- case must be compile-time constant
- Cannot use variables
- Cannot use expressions with variables

Switch-Case - Summary

Key Points:

- Multi-way branching for discrete values
- Works with int and char types
- Cases must be compile-time constants
- break exits the switch
- default handles unmatched values
- Multiple cases can share code (fall-through)
- More readable than many if-else for discrete values

Components:

- **switch(expr)**: Expression to test
- **case value:**: Constant to match
- **break**: Exit switch
- **default**: Optional catch-all

Best Practices

1. **Always use break** unless fall-through intended
2. **Include default** case for error handling
3. **Group related cases** for same action
4. **Comment intentional** fall-through
5. **Use const or #define** for case values
6. **Keep cases simple** - avoid complex logic
7. **Order cases** logically (numeric, alphabetic, frequency)
8. **Use switch** for discrete values, if-else for ranges
9. **Don't modify** switch variable inside cases
10. **Test all paths** including default

Common Pitfalls

1. **Missing break**: Unintended fall-through
2. **No default**: Unhandled values
3. **Variable in case**: Must be constant
4. **Float/double**: Cannot use in switch
5. **String**: Cannot switch on strings in C
6. **Range check**: Use if-else instead
7. **Duplicate cases**: Compiler error
8. **Case outside switch**: Syntax error

Practice Exercises

Try these programs:

1. Month name from number (1-12)
2. Number to word converter (0-9)
3. Traffic light simulator (R/Y/G)
4. Roman numeral converter (I,V,X,L,C,D,M)
5. Shape area calculator (menu-based)
6. Unit converter (length, weight, temp)
7. Grade calculator with letter grades
8. ATM machine simulator
9. File operation menu (create/read/update/delete)
10. Game state machine (start/play/pause/end)