CS 137

Programming Principles

Chapter 2

Making Decisions

Victoria Sakhnini

Table of Contents

1. The if Statement

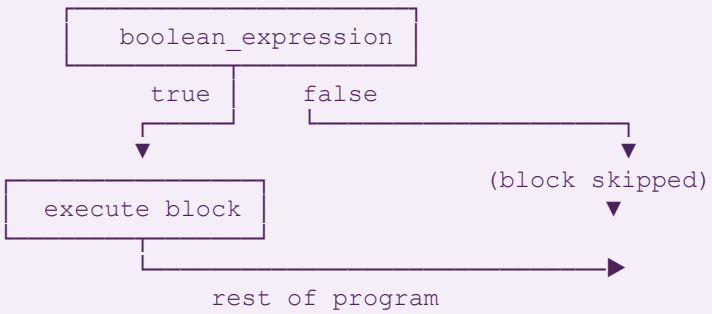
An `if` statement consists of a Boolean expression followed by one or more statements. If the expression is **true**, the block executes; if **false**, the block is skipped.

Syntax:

```
if (boolean_expression) {
    /* statement(s) execute when boolean_expression is true */
}
/* rest of the program continues here regardless */
```

> 📌 C treats any non-zero value as true and zero as false. So if (1), if (42), and if (-5) all enter the block, while if (0) skips it.

Flow:



Example — print the absolute value of a number:

```
#include <stdio.h>

int main(void)
{
    printf("Enter an integer: ");
    int x;
    scanf("%d", &x);
    if (x < 0)
        x *= -1;          // negate if negative
    printf("x = %d\n", x);
}
```

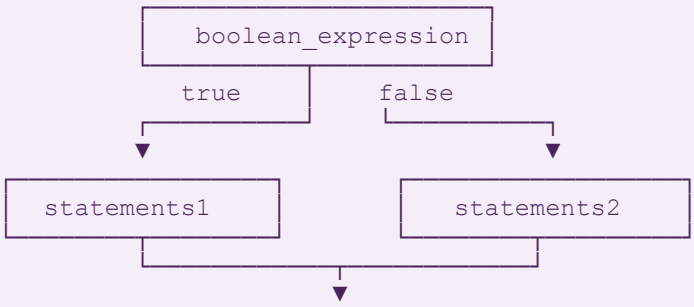
2. The if … else Statement

An `if` statement can be followed by an optional `else` block, which executes when the Boolean expression is **false**.

Syntax:

```
if (boolean_expression) {
    statements1;    /* executes when expression is true  */
}
else {
    statements2;    /* executes when expression is false */
}
/* rest of the program */
```

Flow:



Example — odd or even:

```
#include <stdio.h>

int main(void)
{
    int a;
    printf("Enter an integer:\n");
    scanf("%d", &a);
    if (a % 2 == 0) {
        printf("even\n");
    }
    else {
        printf("odd\n");
    }
    return 0;
}
```

Even when the body of an if or else is a single statement, using curly braces { } is strongly recommended. It prevents bugs when you add more statements later.

3. The if … else if … else Statement

This construct is used to test multiple conditions in sequence. Key rules:

- An `if` can have zero or one `else` — it must come **after** all `else if` branches.
- An `if` can have zero or more `else if` branches — they must come **before** the `else`.
- Once an `else if` condition is true, all remaining `else if` and `else` branches are **skipped**.

Syntax:

```
if (condition1) {
    /* executes when condition1 is true */
} else if (condition2) {
    /* executes when condition1 is false and condition2 is true */
} else if (condition3) {
    /* executes when conditions 1 & 2 are false and condition3 is true */
} else {
    /* executes when none of the above conditions are true */
}
```

Example — letter grade from numerical score:

```
#include <stdio.h>

int main(void)
{
    int  score;
    char grade;
    printf("Enter a numerical grade (0-100):\n");
    scanf("%d", &score);

    if      (score >= 90) grade = 'A';
    else if (score >= 80) grade = 'B';
    else if (score >= 70) grade = 'C';
    else if (score >= 60) grade = 'D';
    else                  grade = 'F';

    printf("The letter grade is: %c\n", grade);
    return 0;
}
```

📌 Use %c as the format specifier to print a single character with printf.

4. Nested if Statements

It is legal in C to nest `if-else` statements — that is, use an `if` or `else if` inside another `if` or `else` block.

Example — compare two integers:

```c
#include <stdio.h>

int main(void)
{
    int var1, var2;
    printf("Input the value of var1: ");
    scanf("%d", &var1);
    printf("Input the value of var2: ");
    scanf("%d", &var2);

    if (var1 != var2) {
        printf("var1 is not equal to var2\n");
        // Nested if-else
        if (var1 > var2)
            printf("var1 is greater than var2\n");
        else
            printf("var2 is greater than var1\n");
    }
    else {
        printf("var1 is equal to var2\n");
    }
    return 0;
}
```

Tricky: The Dangling else Problem — which if does the else belong to?

```c
#include <stdio.h>
int main(void)
{
    int i = 3;
    if (i % 2 == 0)       // outer if: 3 % 2 == 0 is FALSE
        if (i == 0)        //   inner if: never reached
            printf("zero\n");
        else               //   this else belongs to the INNER if
            printf("how odd\n");
    return 0;
}
```

Output: (nothing is printed)

Explanation: The else always belongs to the nearest (innermost) if.
Since i % 2 == 0 is false, the entire inner if-else block is skipped.
Always use braces { } to make nesting explicit and avoid this pitfall!

5. The Ternary Conditional Operator

The ternary operator is a compact, inline way to write a simple if-else expression.

Syntax:

```
result = expr1 ? expr2 : expr3;
```

If `expr1` is true (non-zero), the expression evaluates to `expr2`; otherwise it evaluates to `expr3`.

Example — find the maximum of four integers:

```
#include <stdio.h>

int main(void)
{
    int a, b, c, d, m1, m2;
    printf("Enter four integers: ");
    scanf("%d %d %d %d", &a, &b, &c, &d);

    m1 = a > b ? a : b;              // max of a and b
    m2 = c > d ? c : d;              // max of c and d
    printf("The maximum value: %d\n", m1 > m2 ? m1 : m2);
    return 0;
}
```

Expression	Equivalent if-else	Result when a=5, b=3
m = a>b ? a : b	if(a>b) m=a; else m=b;	m = 5
x = n<0 ? -n : n	if(n<0) x=-n; else x=n;	absolute value of n
s = x%2==0 ? "even" : "odd"	if(x%2==0) s="even"; else s="odd"	depends on x

Ternary operators are great for short, simple conditions. For complex logic with multiple statements, use a regular if-else for clarity.

6. The switch Statement

A `switch` statement tests a variable against a list of constant values (*cases*). It is often cleaner than a long `if-else if` chain when checking one variable against many specific values.

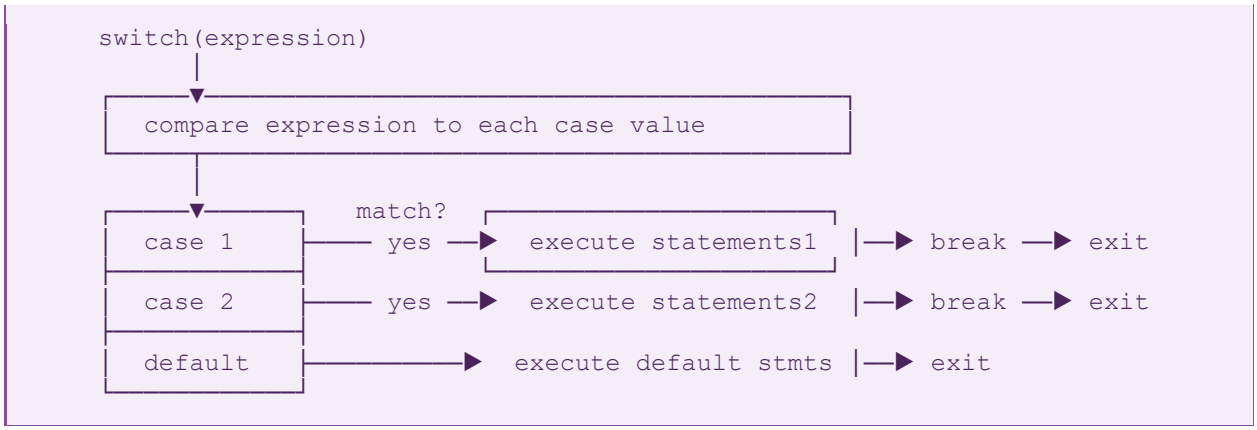
Syntax:

```
switch (expression) {
    case constant1:
        statements1;
        break;          /* optional — exits the switch */

    case constant2:
        statements2;
        break;

    /* ... any number of cases ... */

    default:            /* optional — runs if no case matches */
        statements_n;
}
```

Rules for switch:

- Any number of `case` statements are allowed within a `switch`.
- The constant expression for each `case` must match the type of the `switch` expression.
- When a matching `case` is found, statements execute until a `break` (or the end of the `switch`) is reached.
- Without a `break`, execution **falls through** to the next case automatically.
- The `default` case is optional and runs when no `case` matches. It requires no `break`.

Flow:



Example — letter grade feedback:

```c
#include <stdio.h>

int main(void)
{
    char grade;
    printf("Enter a grade:\n");
    scanf("%c", &grade);

    switch (grade) {
        case 'A':
            printf("Excellent!\n");
            break;
        case 'B':             // fall-through: B and C share the same message
        case 'C':
            printf("Well done\n");
            break;
        case 'D':
            printf("You passed\n");
            break;
        case 'F':
            printf("Better try again\n");
            break;
        default:
            printf("Invalid grade\n");
    }
    printf("Thank you!\n");
    return 0;
}
```

❓ What happens when the input is 'A'?

case 'A' matches → prints "Excellent!" → break exits the switch → prints "Thank you!"

❓ What happens when the input is 'B'?

case 'B' matches — but there is no break, so execution falls through to case 'C'.
→ prints "Well done" → break exits the switch → prints "Thank you!"

This intentional fall-through is used to give B and C the same message.

❓ What happens when the input is a lowercase 'a'?

No case matches (cases are uppercase only) → default executes → prints "Invalid grade"
→ then prints "Thank you!"

📌 switch works with integer types (int, char, long) and enum values. It does NOT work with float, double, or strings.

7. Additional Examples

Example — Compass Heading to Bearing:

This program converts a compass heading in degrees (0–360) to its corresponding bearing description.

Heading Range	Bearing Formula	Example
0 – 89.999°	North (heading) East	45° → "north 45.0 degrees east"
90 – 179.999°	South (180 – heading) East	120° → "south 60.0 degrees east"
180 – 269.999°	South (heading – 180) West	200° → "south 20.0 degrees west"
270 – 360°	North (360 – heading) West	315° → "north 45.0 degrees west"

```c
#include <stdio.h>

int main(void)
{
    double heading;
    printf("Enter a compass heading> ");
    scanf("%lf", &heading);

    if (heading < 0.0)
        printf("Error -- negative heading (%.1f)\n", heading);
    else if (heading < 90.0)
        printf("The bearing is north %.1f degrees east\n", heading);
    else if (heading < 180.0)
        printf("The bearing is south %.1f degrees east\n", 180.0 - heading);
    else if (heading < 270.0)
        printf("The bearing is south %.1f degrees west\n", heading - 180.0);
    else if (heading <= 360.0)
        printf("The bearing is north %.1f degrees west\n", 360.0 - heading);
    else
        printf("Error -- heading > 360 (%.1f)\n", heading);
    return 0;
}
```

8. Extra Practice Problems

Try to solve each problem before looking at the hints or solutions.

Problem 1

> Write a program that reads three different positive integers and prints the middle value.
> Example: input 10 30 12 → output 12 (because 10 < 12 < 30)

Solution:

```
#include <stdio.h>
int main(void) {
    int a, b, c;
    scanf("%d %d %d", &a, &b, &c);
    if      ((a >= b && a <= c) || (a >= c && a <= b)) printf("%d\n", a);
    else if ((b >= a && b <= c) || (b >= c && b <= a)) printf("%d\n", b);
    else                                               printf("%d\n", c);
    return 0;
}
```

Problem 2

a) Solve Problem 1 again without using nested if statements.
b) How many unique test cases do you need to test your solution thoroughly?

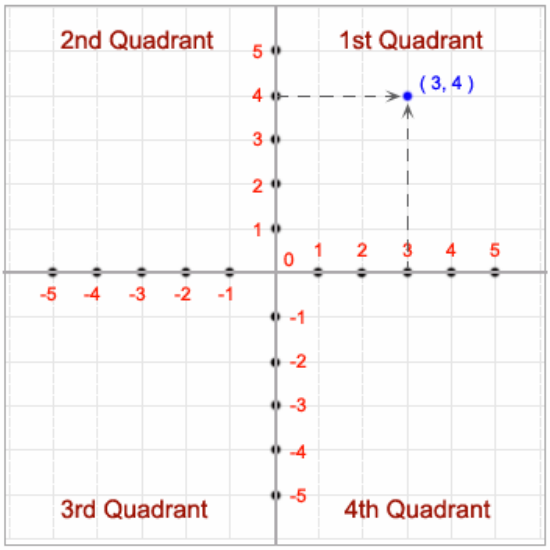
b) 6 test cases — one for each possible ordering of the three values:
a < b < c, a < c < b, b < a < c, b < c < a, c < a < b, c < b < a

Problem 3

Write a program that reads two integers representing coordinates (x, y) and prints which quadrant the point lies in, or whether it is on an axis.
Example: input 3 4 → output "1st quadrant"

Region	Condition	Output
1st quadrant	x > 0 and y > 0	1st quadrant
2nd quadrant	x < 0 and y > 0	2nd quadrant
3rd quadrant	x < 0 and y < 0	3rd quadrant
4th quadrant	x > 0 and y < 0	4th quadrant
Positive x-axis	y == 0 and x > 0	positive x-axis
Origin	x == 0 and y == 0	origin
etc.	...	...

Minimum test cases needed: at least 6
(one for each of the 4 quadrants + one for each axis = 4 + 2 axes, plus origin = 6-9 tests)



Problem 4

Write a program that reads a Celsius temperature and prints a message:
< 0 → Freezing
1 to 10 → Very cold
11 to 20 → Cold
21 to 29 → Nice
30 to 40 → Hot
≥ 40 → Very hot

Problem 5

Write a program (add_check.c) that reads a 4-digit natural number n.
If the digit sum is divisible by 10, the number is valid (print as-is).
Otherwise, append a check digit d such that (digit_sum + d) is divisible by 10.

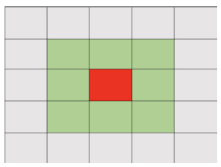
Sample input 1: 8675 → 8+6+7+5 = 26, need 4 more → output: 86754
Sample input 2: 5555 → 5+5+5+5 = 20, divisible by 10 → output: 5555

Problem 6

Yuhan and two friends attend a 5×5 arena mini-meet.
- Edge seats (row/col == 0 or 4): $40 each
- Inner seats (all others): $60 each

Read coordinates for three friends and print the total cost.

Example: seats (0,0), (1,1), (2,1) → 40 + 60 + 60 = 160



Problem 7

Write cinema.c that reads 'age' and 'time' (24-hour) and prints the ticket price:
- Under 13 or over 60: $5
- Everyone else: $10
- Movie between 20:00 and 24:00: $2 discount

Sample Input	Sample Output	Reason
12 15	5	Under 13 → base $5, no evening discount
30 21	8	Adult → $10, evening (21:00) → -$2 = $8
65 19	5	Over 60 → base $5, not evening → $5