3. Decision Making and Branching

Syeda Jannatul Naim
Lecturer | Dept. of CSE
World University of Bangladesh (WUB)

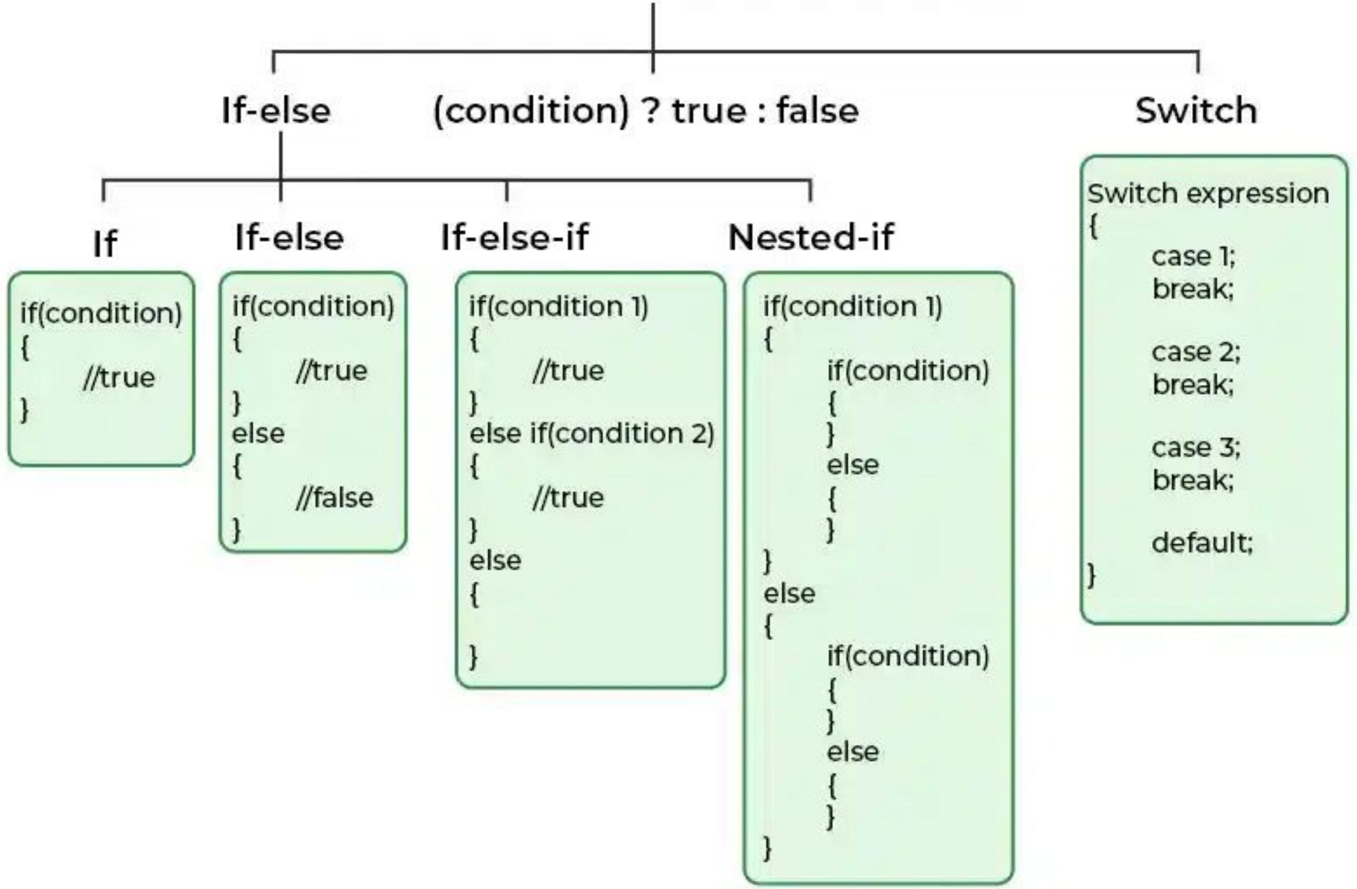
January, 2025

Content

1. Introduction: Decision Making & Branching

- ❑ Decision making allows programs to execute different code blocks based on conditions. It makes programs dynamic and intelligent by enabling them to choose paths based on user input, calculations, or system states.
- ❑ **Decision making** allows a program to:
  - **choose different paths**
  - **execute statements based on conditions**
- ❑ C uses **conditional statements and operators** to control the flow of execution.

Conditional Statements in C



```
if(condition)
{
        //true
}
```

```
if(condition)
{
        //true
}
else
{
        //false
}
```

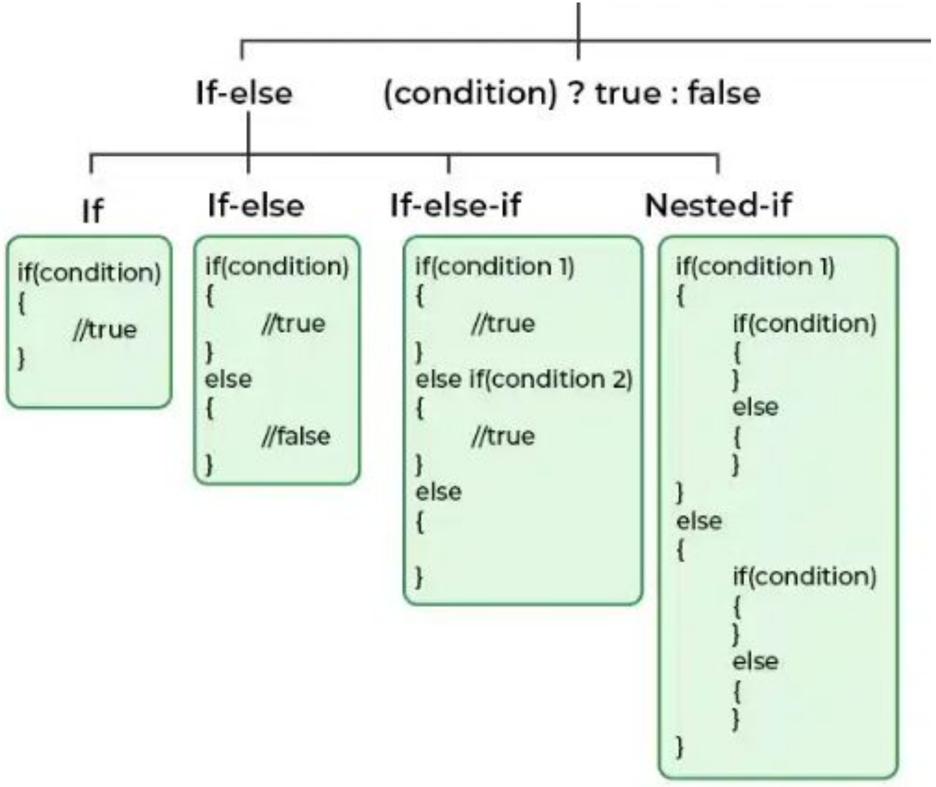
```
if(condition 1)
{
        //true
}
else if(condition 2)
{
        //true
}
else
{

}
```

```
if(condition 1)
{
        if(condition)
        {
        }
        else
        {
        }
}
else
{
        if(condition)
        {
        }
        else
        {
        }
}
```

```
Switch expression
{
        case 1;
        break;

        case 2;
        break;

        case 3;
        break;

        default;
}
```

2. Decision Making with IF Statement

The IF statement is the fundamental building block for decision making. It evaluates a condition and executes code only if the condition is true.



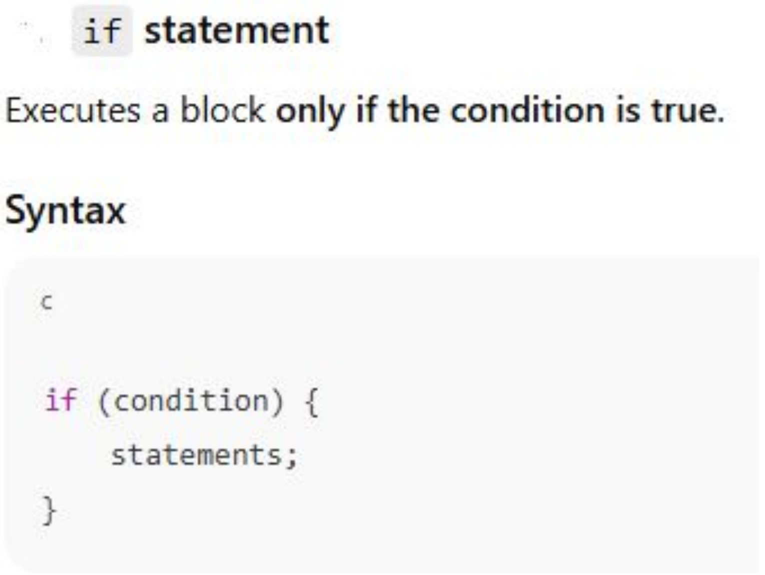
```
if(condition)
{
        //true
}
```

```
if(condition)
{
        //true
}
else
{
        //false
}
```

```
if(condition 1)
{
        //true
}
else if(condition 2)
{
        //true
}
else
{

}
```

```
if(condition 1)
{
        if(condition)
        {
        }
        else
        {
        }
}
else
{
        if(condition)
        {
        }
        else
        {
        }
}
```

2.1 Simple IF Statement



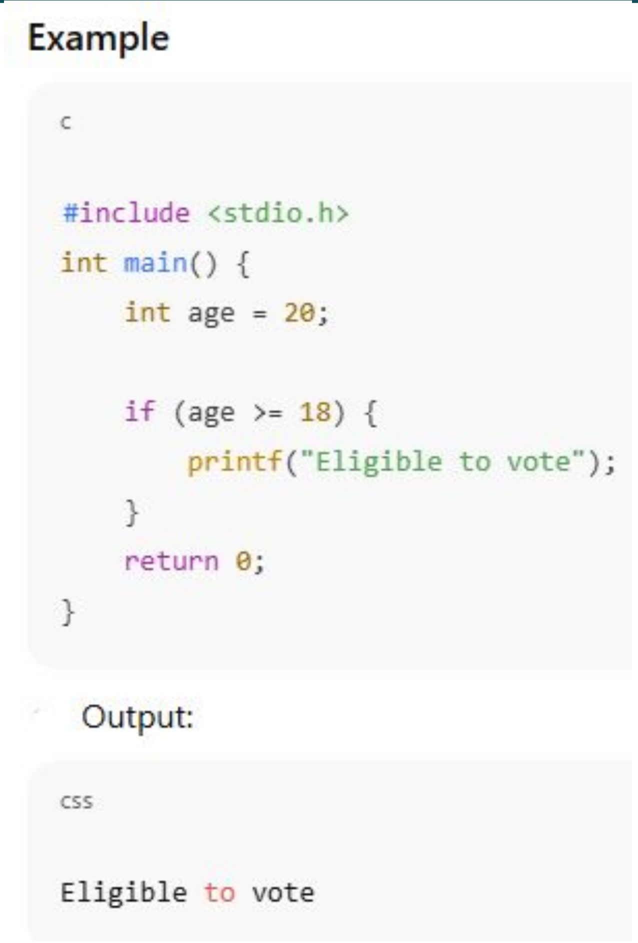
`if` statement

Executes a block **only if the condition is true.**

Syntax

c

```c
if (condition) {
    statements;
}
```



Example

c

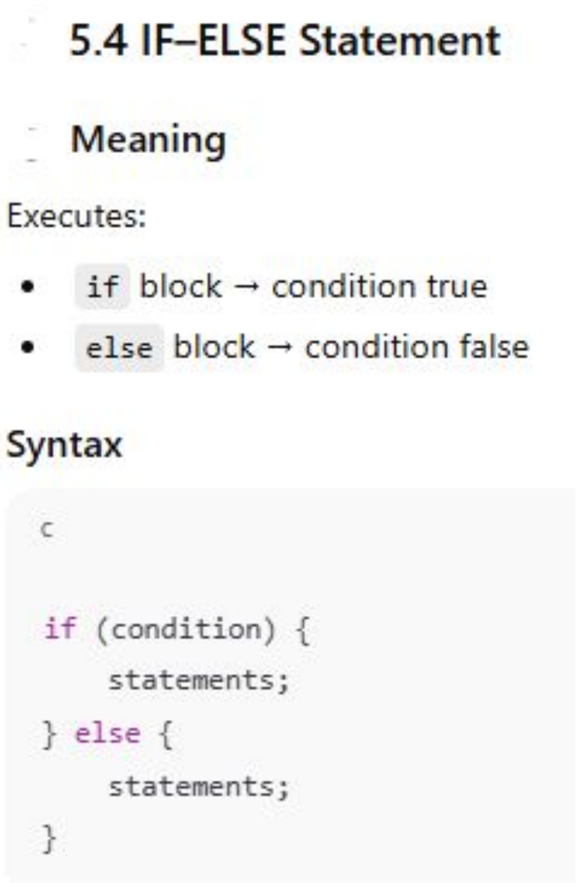
```c
#include <stdio.h>
int main() {
    int age = 20;

    if (age >= 18) {
        printf("Eligible to vote");
    }
    return 0;
}
```

Output:

css

```
Eligible to vote
```

2.2 IF-ElSE Statement



5.4 IF–ELSE Statement

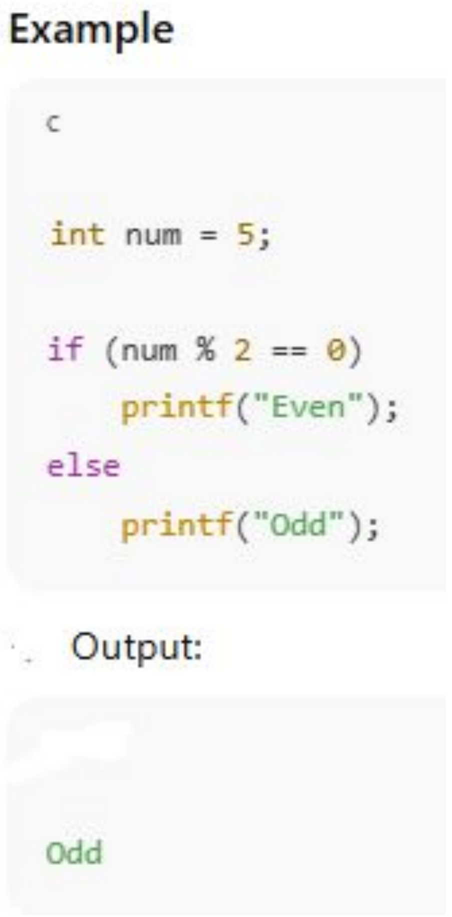
Meaning

Executes:

- `if` block → condition true
- `else` block → condition false

Syntax

```c
if (condition) {
    statements;
} else {
    statements;
}
```



Example

```c
int num = 5;

if (num % 2 == 0)
    printf("Even");
else
    printf("Odd");
```

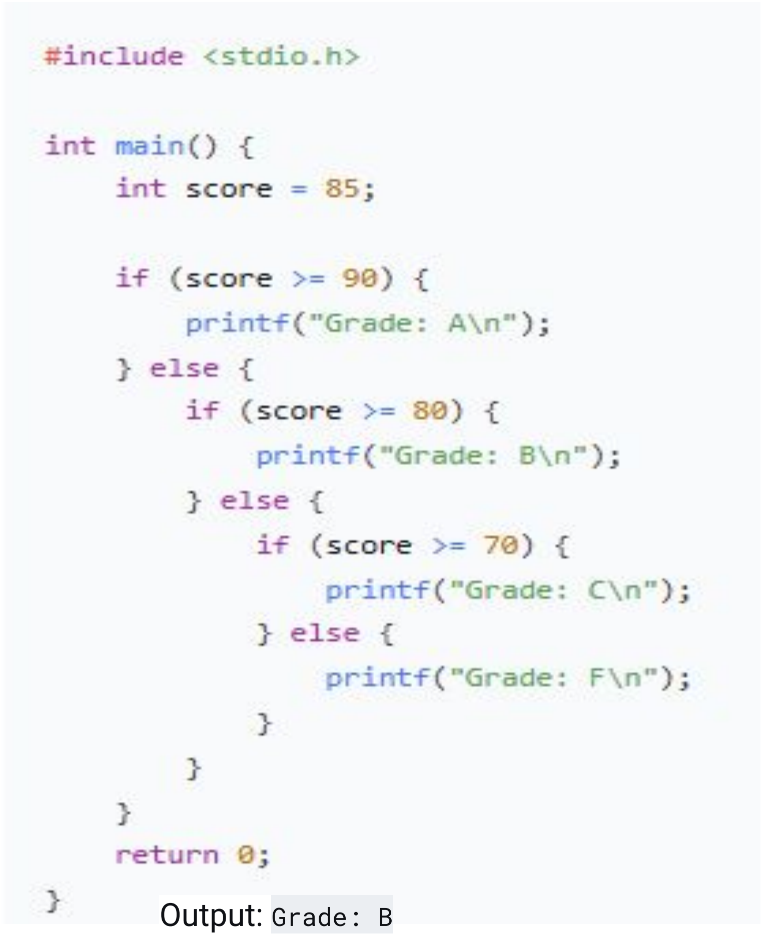
Output:

```
Odd
```

2.3 Nesting of IF–ELSE Statement

Placing one IF-ELSE statement inside another.

Example: Grade Classification



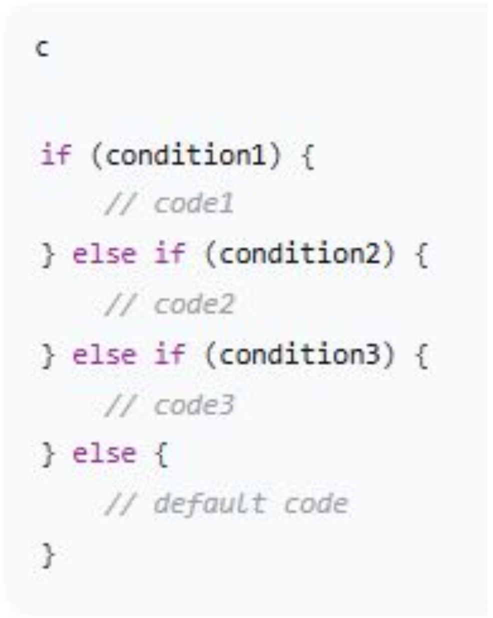
```c
#include <stdio.h>

int main() {
    int score = 85;

    if (score >= 90) {
        printf("Grade: A\n");
    } else {
        if (score >= 80) {
            printf("Grade: B\n");
        } else {
            if (score >= 70) {
                printf("Grade: C\n");
            } else {
                printf("Grade: F\n");
            }
        }
    }
    return 0;
}
```

Output: `Grade: B`

2.4 The ELSE-IF Ladder

A cleaner way to handle multiple conditions sequentially.

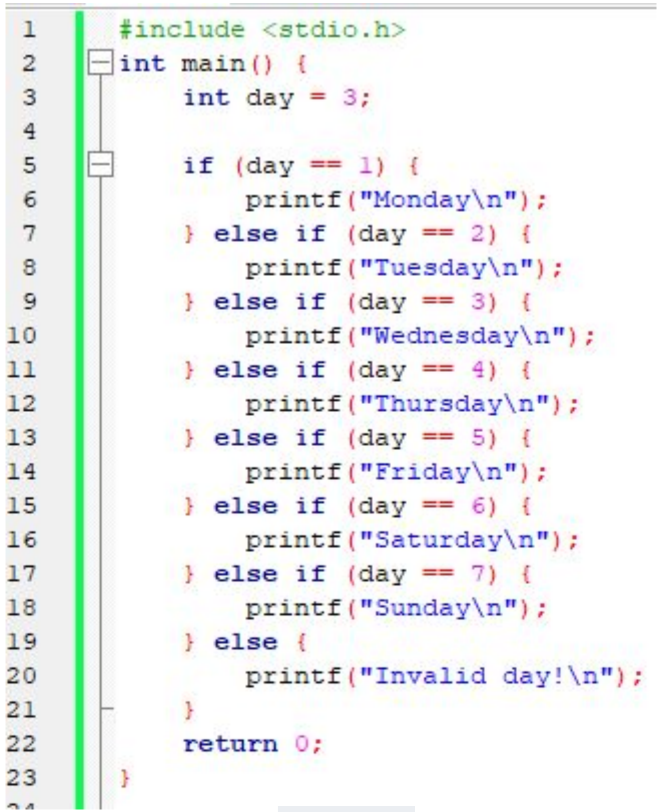
Syntax:



c

```c
if (condition1) {
    // code1
} else if (condition2) {
    // code2
} else if (condition3) {
    // code3
} else {
    // default code
}
```

Example: Day of Week



```c
#include <stdio.h>
int main() {
    int day = 3;

    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: `Wednesday`

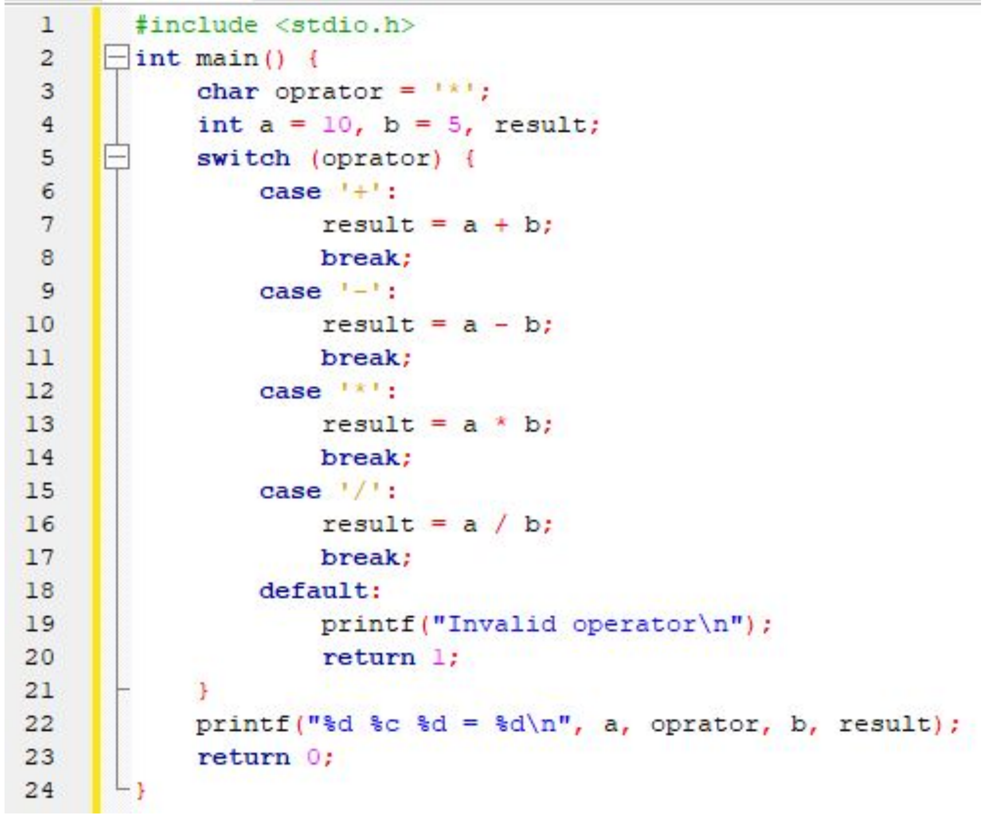
3. Switch Statement

Used when a variable is compared with multiple constant values.Efficient for multiple choices based on a single variable/value.

Syntax:

```
switch (expression) {
    case value1:
        statements;
        break;
    case value2:
        statements;
        break;
    default:
        statements;
}
```

Example: Calculator



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

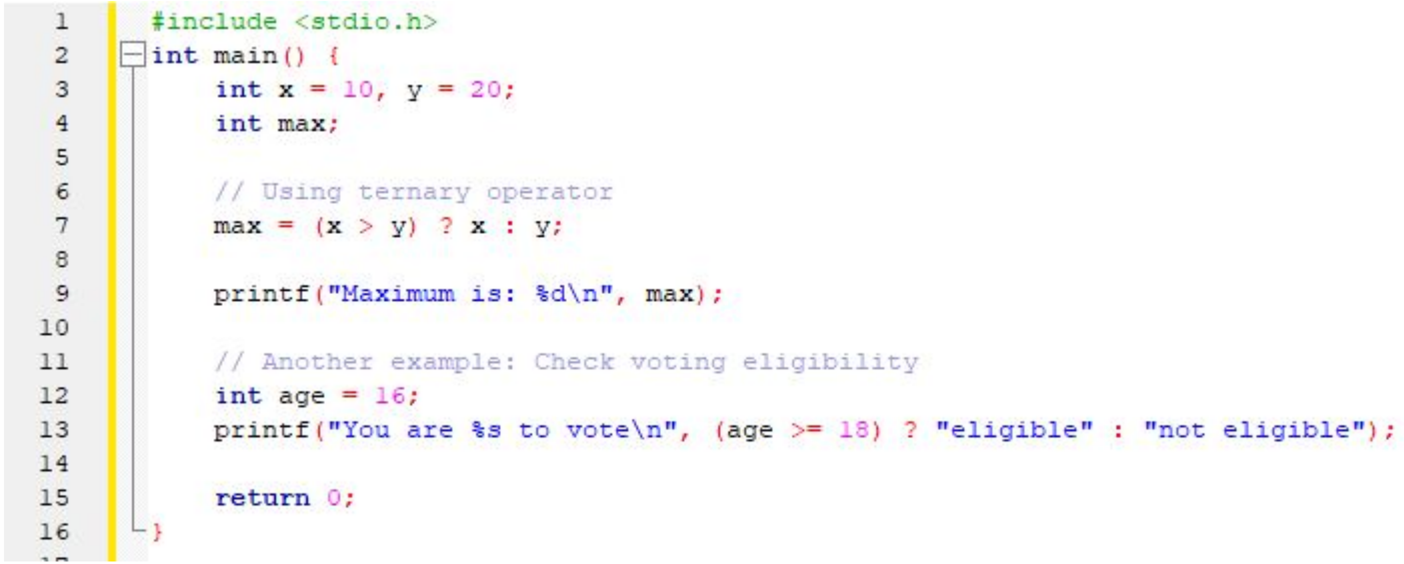
Output: `10 * 5 = 50`

4. The ?: (Conditional / Ternary Operator)

A shorthand for simple IF-ELSE statements. **Syntax:**

condition ? expression_if_true : expression_if_false;

Example: Find Maximum



```c
#include <stdio.h>
int main() {
    int x = 10, y = 20;
    int max;

    // Using ternary operator
    max = (x > y) ? x : y;

    printf("Maximum is: %d\n", max);

    // Another example: Check voting eligibility
    int age = 16;
    printf("You are %s to vote\n", (age >= 18) ? "eligible" : "not eligible");

    return 0;
}
```

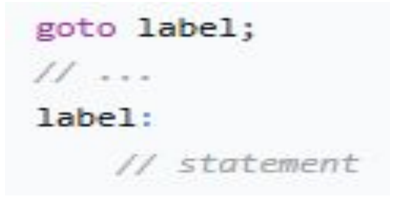
Output:

```
Maximum is: 20
You are not eligible to vote
```

5. The GOTO Statement

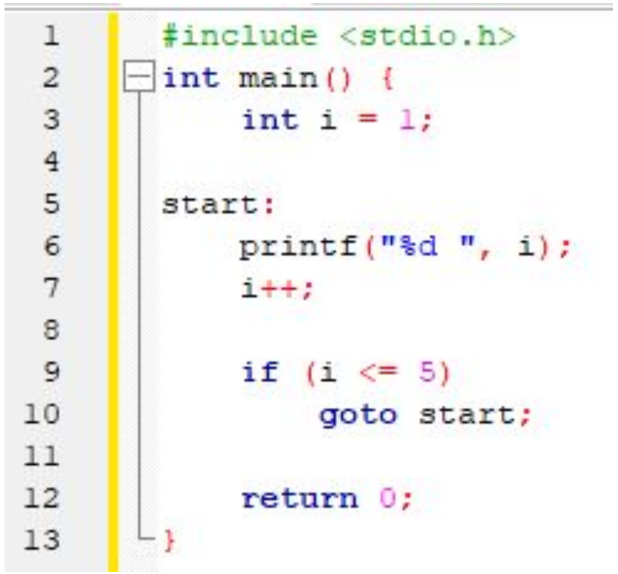
Transfers control unconditionally to a labeled statement. Generally not recommended (makes code confusing).

Syntax:



```
goto label;
// ...
label:
    // statement
```

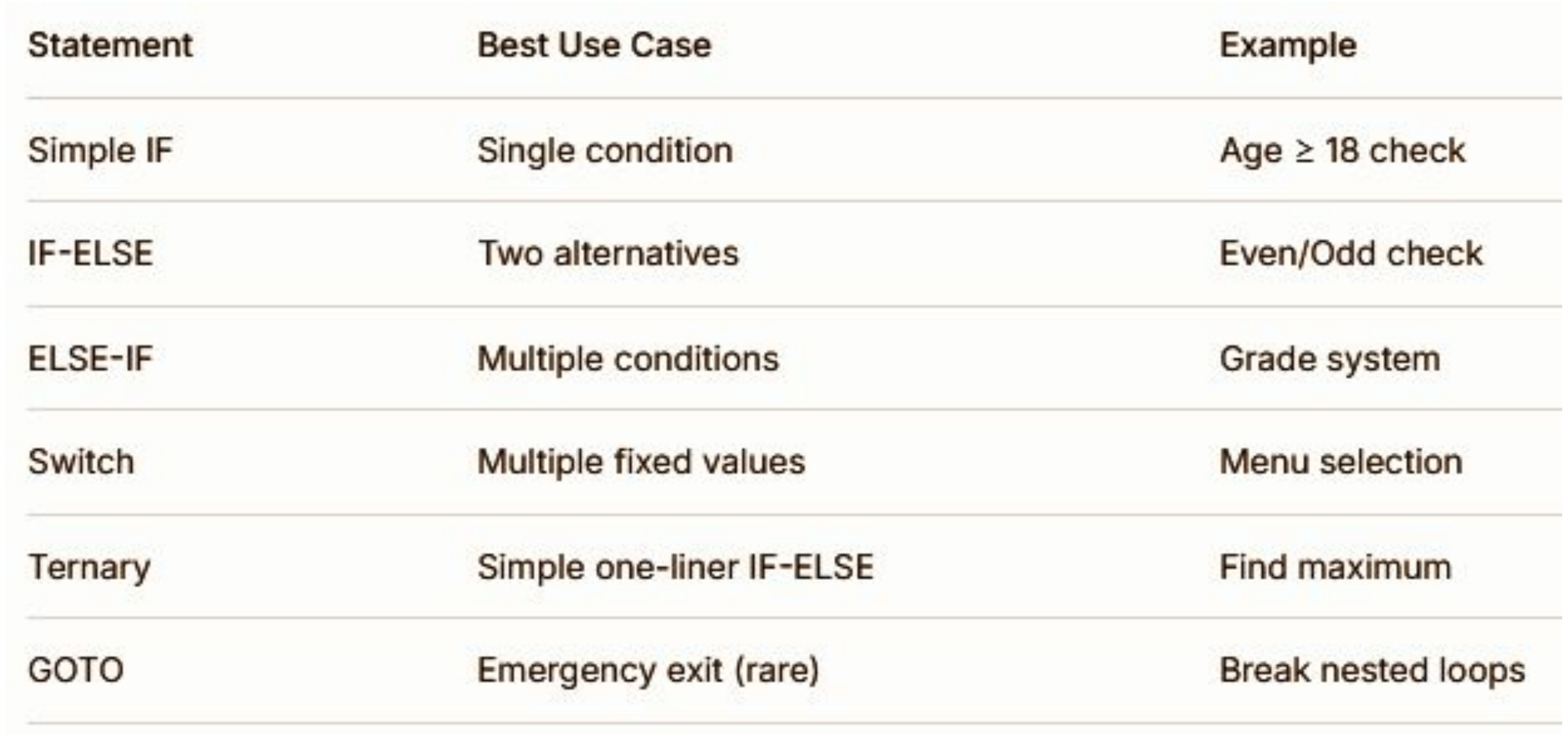
Example: demo of goto



```
#include <stdio.h>
int main() {
    int i = 1;

start:
    printf("%d ", i);
    i++;

    if (i <= 5)
        goto start;

    return 0;
}
```

Output: 1 2 3 4 5

Summary



Statement	Best Use Case	Example
Simple IF	Single condition	Age ≥ 18 check
IF-ELSE	Two alternatives	Even/Odd check
ELSE-IF	Multiple conditions	Grade system
Switch	Multiple fixed values	Menu selection
Ternary	Simple one-liner IF-ELSE	Find maximum
GOTO	Emergency exit (rare)	Break nested loops

END

of

Chapter 3

***Reference**: E. Balaguruswamy; Programming in ANSI C; chapter-5;*