

الرقم المأجوز : ٢٠٢١١٨٢٧

- Q1 / *Advantages: thanks of organizing the code.
- 1. Clearly indicates where a block ends, especially in nested code.
 - 2. Helps with code organization and readability.
 - 3. Familiar to programmers coming from C-like languages.

*Disadvantages:

- 1. It's easy to forget or misplace, which can lead to syntax errors.
- 2. Adds visual clutter, especially in poorly-formatted code.

Q2 / Multi-line Comments: no brackets used.

*Advantages:

- 1. Useful for documenting a long explanations or disabling blocks of code.

*Disadvantages:

- 1. If the closing delimiter is forgotten, it can comment out unintended code.
- 2. Harder to nest or manage in some languages.

11 Single-line Comments:

* Advantages:

1. Quick and simple for short notes or inline explanations
2. easier to use and less error-prone

* Disadvantages:

1. Repetitive if used for multiple lines.
2. Not suitable for large documentation blocks

Q3

- Dynamic type binding means that the type of variable is determined at runtime implicitly. Variables are also created at runtime, and their type and storage are both decided when the variable is assigned a value.

So, both depend on runtime behavior, and both allow more flexibility in programming.

Q4

Visible in sub1 printing numbers not integers.

a - declared in sub1 & global variables X

y - declared in sub1 (hides global y)

z - declared in sub1 (hides global z) X

Visible in sub2 printing numbers not integers.

a - declared in sub2 & declared in sub3 X

b - declared in sub2 X

z - declared in sub2 X

y - from sub1 X

x - from global scope X

a - declared in sub3 X

x - declared in sub3 X

w - declared in sub3 X

y - from global scope X

z - from global scope X

17 (2)

```
main.c
1 #include <stdio.h>
2
3 void fun(void) {
4     int a = 10, b = 20, c = 30;
5     printf("→ 4: a=%d (1), b=%d (1), c=%d (1)\n", a, b, c);
6
7     while (1) {
8         int b = 200, c = 300, d = 400;
9         printf("→ 1: a=%d (1), b=%d (2), c=%d (2), d=%d (2)\n", a, b, c, d);
10    }
11
12    while (1) {
13        int c = 3000, d = 4000, e = 5000;
14        printf("→ 2: a=%d (1), b=%d (2), c=%d (3), d=%d (3), e=%d (3)\n", a, b, c, d, e);
15        break;
16    }
17    printf("→ 3: a=%d (1), b=%d (2), c=%d (2), d=%d (2)\n", a, b, c, d);
18    break;
19 }
20
21
22 int main() {
23     fun();
24     return 0;
25 }
26
```

Output

```
→ 4: a=10 (1), b=20 (1), c=30 (1)
→ 1: a=10 (1), b=200 (2), c=300 (2), d=400 (2)
→ 2: a=10 (1), b=200 (2), c=3000 (3), d=4000 (3), e=5000 (3)
→ 3: a=10 (1), b=200 (2), c=300 (2), d=400 (2)

*** Code Execution Successful ***
```

The screenshot shows a browser's developer tools with the "Console" tab selected. The console output displays a series of log statements from three nested functions: sub2, sub1, and sub3. The log entries show variable assignments and their values. To the right of the log entries, the corresponding line numbers in the source code are listed.

Log Statement	Line Number
sub2:	main.js:134
a: undefined	main.js:135
b: undefined	main.js:136
z: undefined	main.js:137
y: undefined	main.js:138
x: 1	main.js:139
sub1:	main.js:142
a: undefined	main.js:143
y: undefined	main.js:144
z: undefined	main.js:145
x: 1	main.js:146
sub3:	main.js:151
a: undefined	main.js:152
x: undefined	main.js:153
w: undefined	main.js:154

```
var x, y, z;

function sub1() {
    var a, y, z;
    function sub2() {
        var a, b, z;
        console.log("sub2:");
        console.log("a:", a);
        console.log("b:", b);
        console.log("z:", z);
        console.log("y:", y);
        console.log("x:", x);
    }
    sub2();
    console.log("sub1:");
    console.log("a:", a);
    console.log("y:", y);
    console.log("z:", z);
    console.log("x:", x);
}

function sub3() {
    var a, x, w;
    console.log("sub3:");
    console.log("a:", a);
    console.log("x:", x);
    console.log("w:", w);
}

x = 1;
sub1();
sub3();
```

main.cpp

```
1 #include <stdio.h>
2 void fun3(int a, int b, int c, int d, int e, int f) {
3     printf("fun3: a=%d (main), b=%d, c=%d, d=%d, e=%d, f=%d\n", a, b, c, d, e, f);
4 }
5 void fun2(int a, int b, int c, int d, int e) {
6     int f = 60;
7     fun3(a, b, c, d, e, f);
8 }
9
10 void fun1(int a, int b, int c) {
11     int d = 40;
12     int e = 50;
13     fun2(a, b, c, d, e);
14
15     int f = 60;
16     fun3(a, b, c, d, 0, f);
17 }
18
19 int main() {
20     int a = 10, b = 20, c = 30;
21
22     printf("----- Case A: main → fun1 → fun2 → fun3 -----\\n");
23     fun1(a, b, c);
24
25     printf("----- Case B: main → fun1 → fun3 -----\\n");
26     int d = 40, f = 60;
27     fun3(a, b, c, d, 0, f);
28
29     printf("----- Case C: main → fun2 → fun3 → fun1 -----\\n");
30     int e = 50;
31     fun3(a, b, c);
32
33 }
```

Output

```
----- Case A: main → fun1 → fun2 → fun3 -----
fun3: a=10 (main), b=20, c=30, d=40, e=50, f=60
fun3: a=10 (main), b=20, c=30, d=40, e=60, f=60

----- Case B: main → fun1 → fun3 -----
fun3: a=10 (main), b=20, c=30, d=40, e=50, f=60
fun3: a=10 (main), b=20, c=30, d=40, e=0, f=60

----- Case C: main → fun2 → fun3 → fun1 -----
fun3: a=10 (main), b=20, c=30, d=40, e=50, f=60
fun3: a=10 (main), b=20, c=30, d=40, e=0, f=60

*** Code Execution Successful ***
```

٩٦

النص من ملف ...PDF