

الرقم الجامعي : 20211827

الأستاذ: الدكتور محمد بن محمد

Q1/ * Advantages:

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-link

or other languages. Special not relative to the code.

* Disadvantages:

1. Easy to forget or misplace, with some compilers lead to syntax errors.
2. Adds visual clutter, especially in poorly formatted code.

Q2/ * Multi-line Comments

* 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.

14 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. Implicit heap-dynamic 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

Variable in sub 1

a - declared in sub 1

y - declared in sub 1 (hides global y)

z - declared in sub 1 (hides global z)

x - from global scope

Variable in sub 2

a - declared in sub 2

b - declared in sub 2

z - declared in sub 2

y - from sub 1

x - from global scope

Variable in sub 3

a - declared in sub 3

x - declared in sub 3

w - declared in sub 3

y - from global scope

z - from global scope

```
main.c
1 #include <stdio.h>
2
3 void fun(void) {
4     int a = 10, b = 20, c = 30;
5     printf("1: 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("2: a=%d (1), b=%d (2), c=%d (2), d=%d (2)\n", a, b, c, d);
10
11         while (1) {
12             int e = 3000, d = 4000, e = 5000;
13             printf("3: a=%d (1), b=%d (2), c=%d (3), d=%d (3), e=%d (3)\n", a, b, c, d, e);
14             break;
15         }
16
17         printf("4: 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 ===
```



```
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("\n----- Case B: main -> fun1 -> fun3 -----\n");
26     int d = 40, f = 60;
27     fun3(a, b, c, d, 0, f);
28
29     printf("\n----- Case C: main -> fun2 -> fun3 -> fun1 -----\n");
30     int e = 50;
31     fun1(a, b, c);
32     return 0;
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=0, f=60

----- Case B: main -> fun1 -> fun3 -----
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 ===
```

