

## Circle the Variables' Scopes

```
#include <iostream>
using namespace std;
```

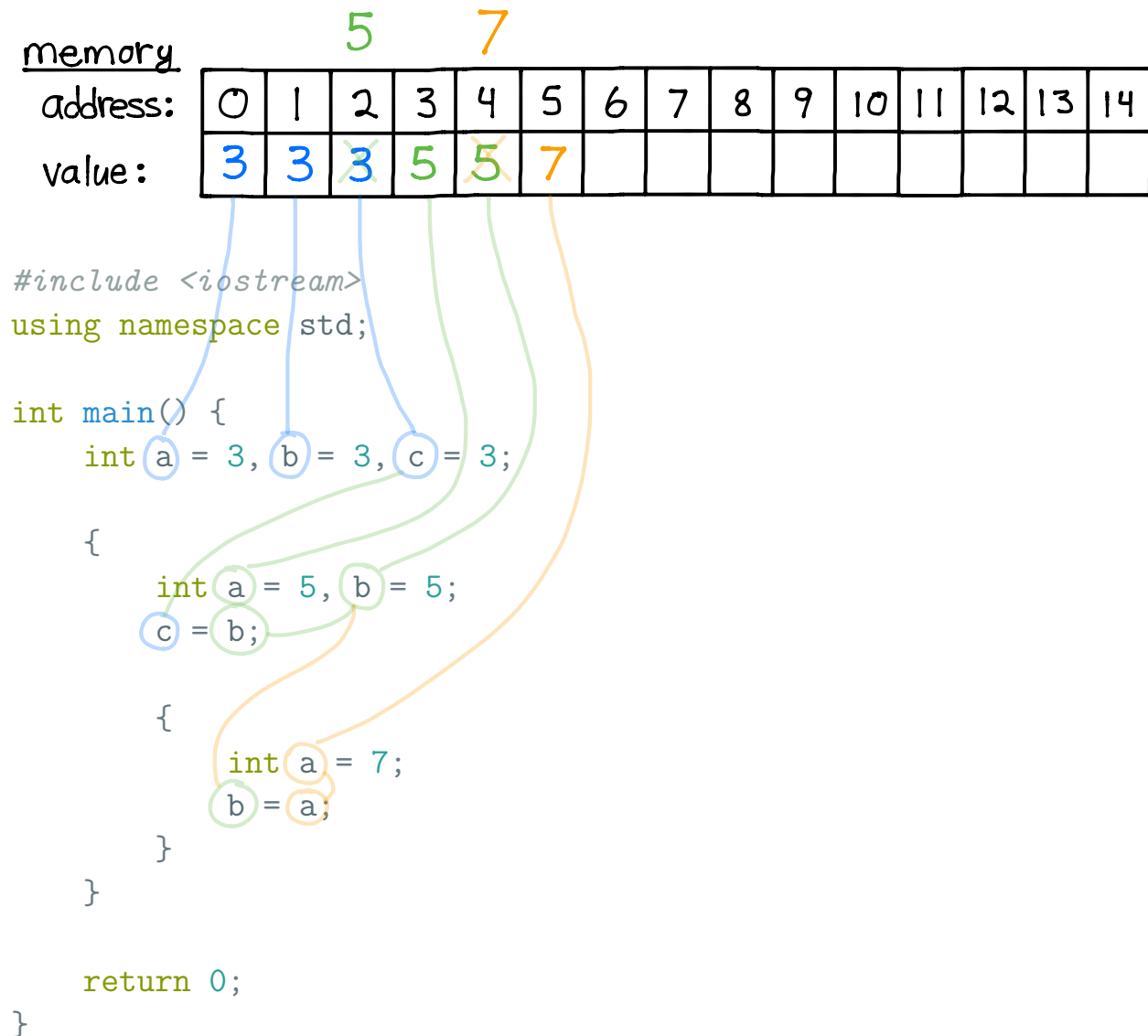
```
int main() {
    int a = 3, b = 3, c = 3;

    {
        int a = 5, b = 5;
        c = b;

        {
            int a = 7;
            b = a;
        }
    }

    return 0;
}
```

## Fill in the Memory



## Circle the Variables' Scopes

```
#include <iostream>
using namespace std;
```

```
int main() {
    int a = 3;

    for (int i = 7; i > a; i--) {
        for (int j = 11; j > i; j--) {
            cout << "*";
        }

        cout << endl;
    }

    return 0;
}
```

The diagram illustrates the scope of variables in the provided C++ code. A large blue circle encompasses the entire `main` function, from `int main()` to the final closing brace. Inside this, a green circle outlines the scope of the `for` loop over `i`, starting from `int i = 7` and ending at the closing brace of the loop. Within the green circle, an orange circle outlines the scope of the nested `for` loop over `j`, starting from `int j = 11` and ending at the closing brace of the inner loop. Individual variables `a`, `i`, and `j` are also circled in blue, green, and orange respectively, with lines connecting these circles to their corresponding scope boundaries. Specifically, a blue line connects the circled `a` to the `i > a` condition, a green line connects the circled `i` to the `j > i` condition, and an orange line connects the circled `j` to the `j--` decrement operation.

```
#include <iostream>
using namespace std;
```

```
int main() {
    int a = 3;
    int i = 7;
    while (i > a) {
        int j = 11;
        while (j > i) {
            cout << "*";
            j--;
        }
        cout << endl;
        i--;
    }
    return 0;
}
```

The diagram illustrates the scope of variables in the provided C++ code. Three nested regions are defined by colored lines: a blue line for the `main()` function, a green line for the `while (i > a)` loop, and an orange line for the inner `while (j > i)` loop. Variables are circled in the same color as their containing scope: `a` is blue, `i` is green, and `j` is orange. Lines connect each variable to its corresponding scope boundary. For example, a blue line connects `a` to the `main()` block, a green line connects `i` to the `while (i > a)` block, and an orange line connects `j` to the inner `while (j > i)` block.

```
#include <iostream>
using namespace std;

int main() {
    // have to be a little careful with do-while loops:
    // it's not always possible to write one that means the
    // same thing as the corresponding for or while loop
```

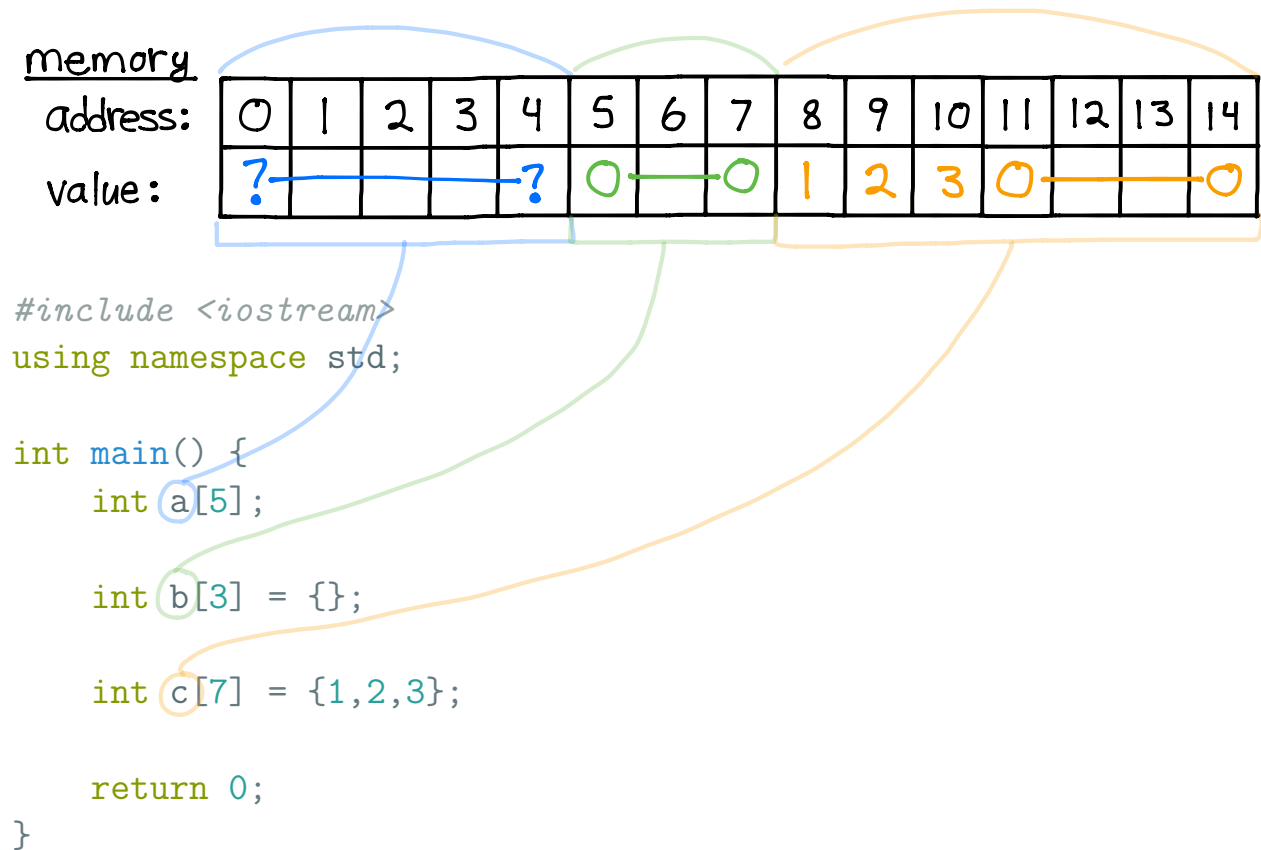
```
int a = 3;
do {
    int i = 7;
    do {
        int j = 11;
        do {
            cout << "*";
            j--;
        } while (j > i);
        cout << endl;
        i--;
    } while (i > a);
} while (i > a);
return 0;
```

The code is annotated with hand-drawn colored boxes and circles to indicate variable scopes:

- A blue box encloses the entire `main` function body.
- A green box encloses the `do` loop starting with `int i = 7;`.
- An orange box encloses the inner `do` loop starting with `int j = 11;`.
- Individual circles highlight the variables `a`, `i`, and `j` in their respective lines of declaration and use within the loops.

## 6 Arrays

### Fill in the Memory



memory

address:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0					0	1	2	3	4	5	6			

value:

```
#include <iostream>
using namespace std;
```

```
int main() {
    int a[2][3] = {};

    int b[2][3] = {1, 2, 3, 4, 5, 6};

    return 0;
}
```

memory

address:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
value:	1	2	3	4	5	6	1	2	0-0	3	4	0-0			

```
#include <iostream>
using namespace std;
```

```
int main() {
    int a[2][3] = { {1, 2, 3}, {4, 5, 6} };

    int b[2][4] = { {1, 2}, {3, 4} };

    return 0;
}
```



memory

address:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	2	3	4	5	6	7	8	9	0	1	2			

value:

```
#include <iostream>
```

```
using namespace std;
```

```
int main() {
```

```
    int a[2][3][2] = { { {1, 2}, {3, 4}, {5, 6} },  
                        { {7, 8}, {9, 0}, {1, 2} } };
```

```
    return 0;
```

```
}
```

memory

address:

The diagram shows a memory layout with two rows: 'address' and 'value'. The 'address' row contains indices from 0 to 14. The 'value' row contains values: 1 at address 0, 0 at address 1, 0 at address 5, 2 at address 6, 0 at address 7, and 0 at address 11. All other cells are empty. Handwritten blue annotations include: arcs connecting address 0 to 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, and 11; arcs connecting address 1 to 2, 3, 4, 5, 6, 7, 8, 9, 10, and 11; a horizontal line connecting the 0 values at addresses 1, 5, 7, and 11; and a bracket under the first six cells of the value row (addresses 0 to 5).

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	0				0	2	0				0			

value:

```
#include <iostream>
```

```
using namespace std;
```

```
int main() {
```

```
    int a[2][3][2] = { {1}, {2} };
```

```
    return 0;
```

```
}
```