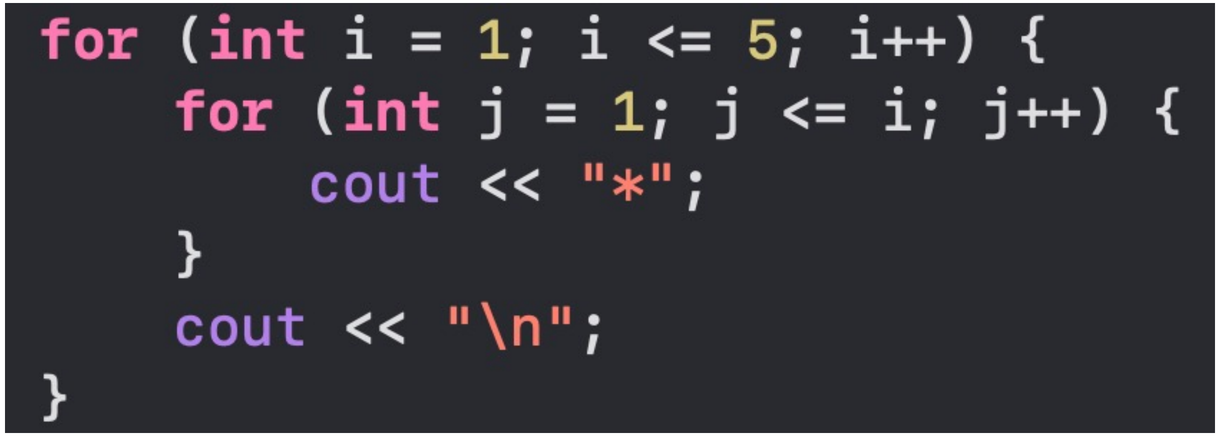
C++基礎語法 Unit-6

- 多重迴圈
- 多維陣列

多重迴圈

多重迴圈 (nested loop)

Example 6-2
(Quiz)



```cpp
for (int i = 1; i <= 5; i++) {
    for (int j = 1; j <= i; j++) {
        cout << "*";
    }
    cout << "\n";
}
```

外圈 i：1 ~ 5
內圈 j：1 ~ i

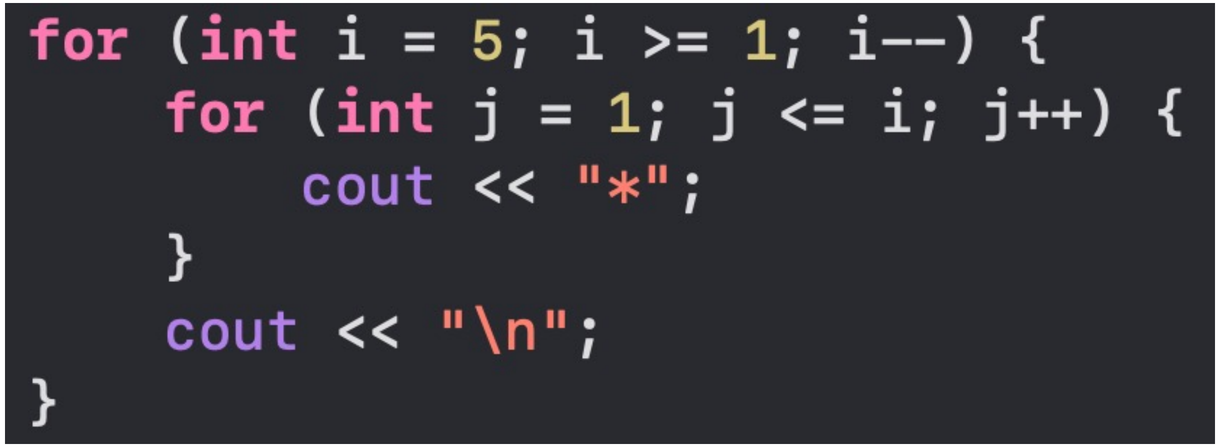
```
    *
   ***
  *****
 *******
*********
```

```
*
**
***
****
*****
```

```
*****
****
***
**
*
```

多重迴圈 (nested loop)

Example 6-2
(Quiz)



```
for (int i = 5; i >= 1; i--) {
    for (int j = 1; j <= i; j++) {
        cout << "*";
    }
    cout << "\n";
}
```

外圈 i：5 ~ 1
內圈 j：1 ~ i

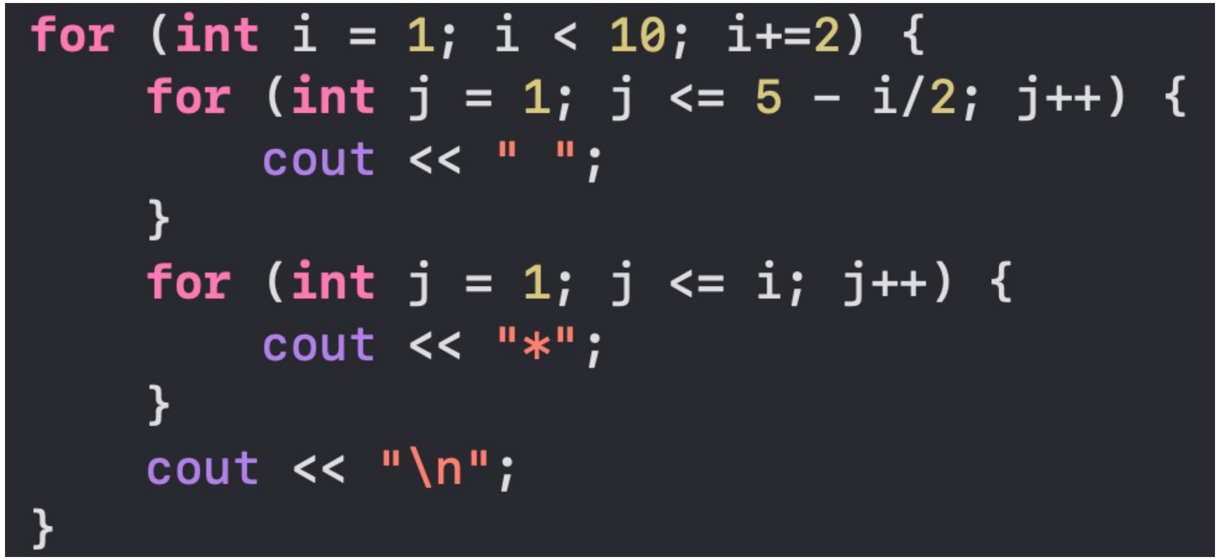
```
    *
   ***
  *****
 *******
*********
```

```
*
**
***
****
*****
```

```
*****
****
***
**
*
```

多重迴圈 (nested loop)

Example 6-2
(Quiz)



```
for (int i = 1; i < 10; i+=2) {
    for (int j = 1; j <= 5 - i/2; j++) {
        cout << " ";
    }
    for (int j = 1; j <= i; j++) {
        cout << "*";
    }
    cout << "\n";
}
```

外圈 i：1, 3, 5, 7, 9
內圈 j：1 ~ 5 – i/2

內圈 j：1 ~ i

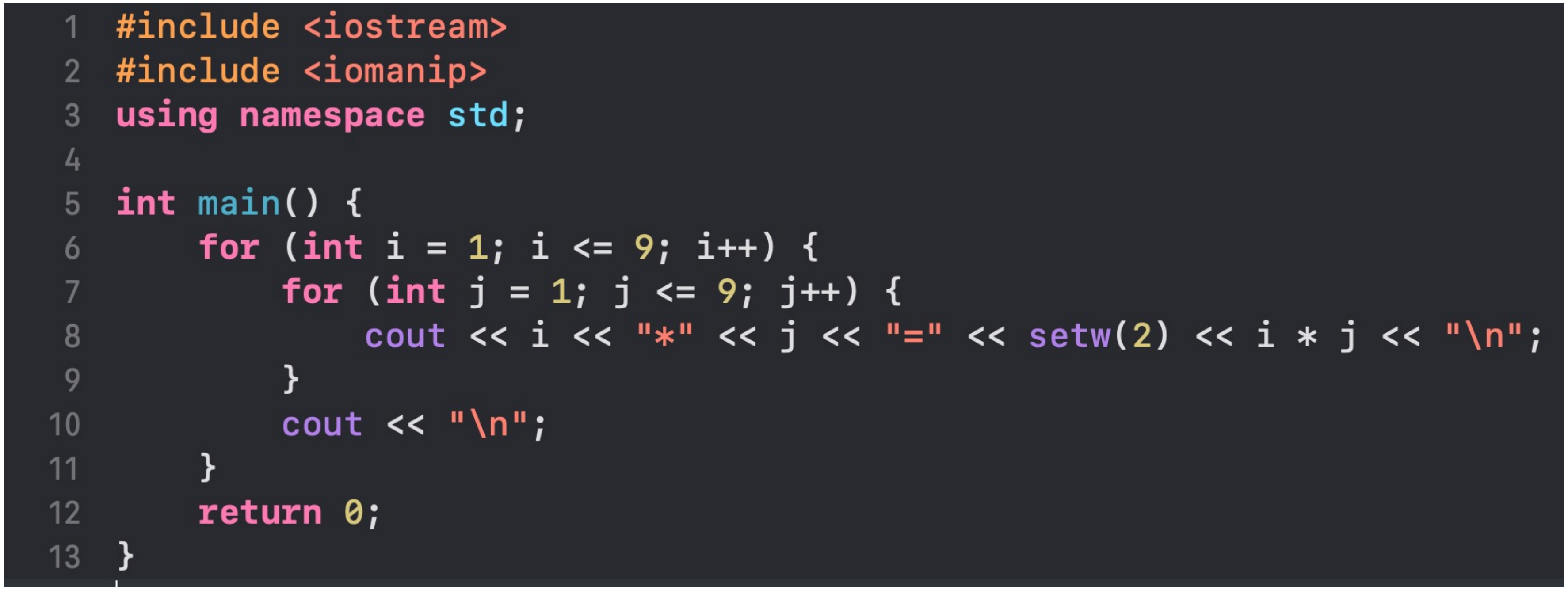
```
    *
   ***
  *****
 *******
*********
```

```
*
**
***
****
*****
```

```
*****
****
***
**
*
```

【範例】九九乘法表

Example 6-3



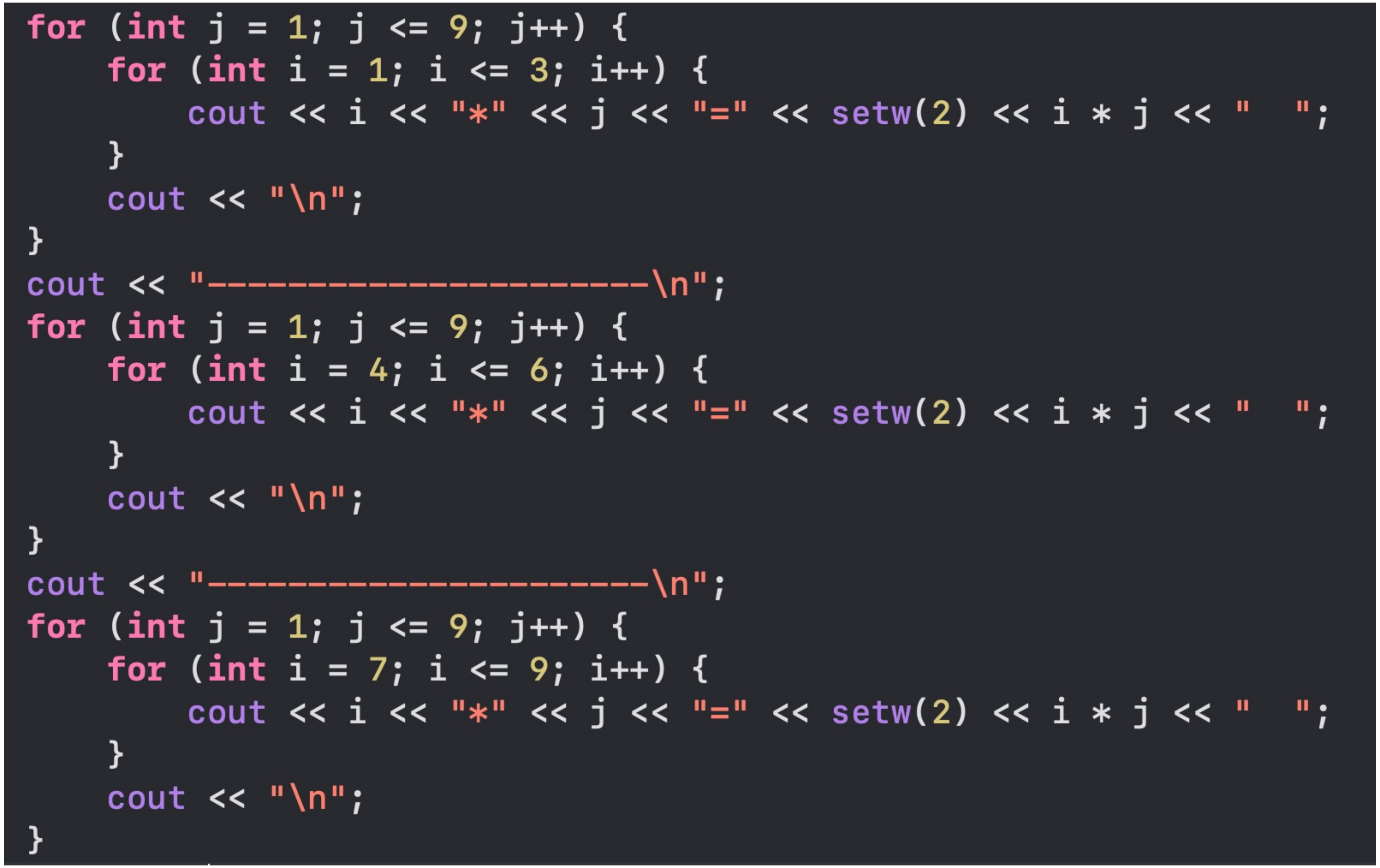
```
#include <iostream>
#include <iomanip>
using namespace std;

int main() {
    for (int i = 1; i <= 9; i++) {
        for (int j = 1; j <= 9; j++) {
            cout << i << "*" << j << "=" << setw(2) << i * j << "\n";
        }
        cout << "\n";
    }
    return 0;
}
```

外圈 i：1 ～ 9
內圈 j：1 ～ 9
i * j

Example 6-4

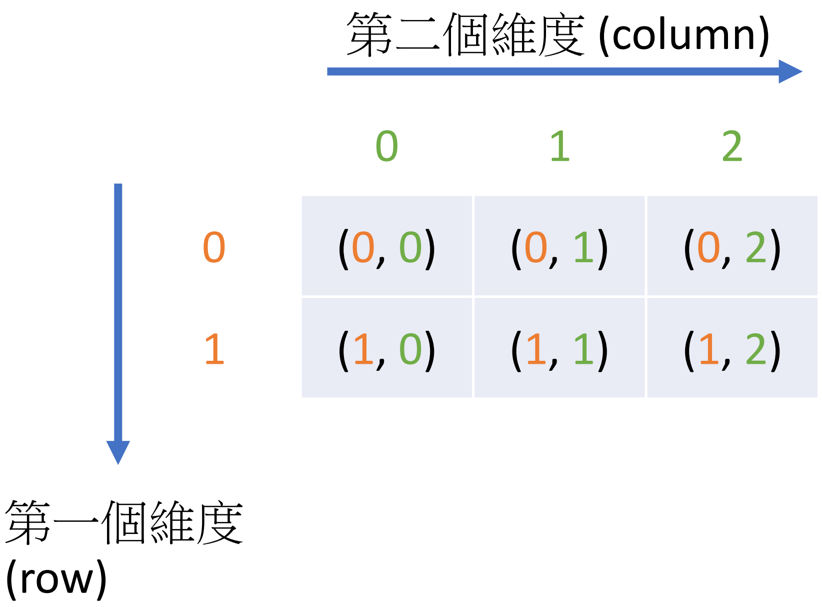
```
1*1= 1  2*1= 2  3*1= 3
1*2= 2  2*2= 4  3*2= 6
1*3= 3  2*3= 6  3*3= 9
1*4= 4  2*4= 8  3*4=12
1*5= 5  2*5=10  3*5=15
1*6= 6  2*6=12  3*6=18
1*7= 7  2*7=14  3*7=21
1*8= 8  2*8=16  3*8=24
1*9= 9  2*9=18  3*9=27
----------------------
4*1= 4  5*1= 5  6*1= 6
4*2= 8  5*2=10  6*2=12
4*3=12  5*3=15  6*3=18
4*4=16  5*4=20  6*4=24
4*5=20  5*5=25  6*5=30
4*6=24  5*6=30  6*6=36
4*7=28  5*7=35  6*7=42
4*8=32  5*8=40  6*8=48
4*9=36  5*9=45  6*9=54
----------------------
7*1= 7  8*1= 8  9*1= 9
7*2=14  8*2=16  9*2=18
7*3=21  8*3=24  9*3=27
7*4=28  8*4=32  9*4=36
7*5=35  8*5=40  9*5=45
7*6=42  8*6=48  9*6=54
7*7=49  8*7=56  9*7=63
7*8=56  8*8=64  9*8=72
7*9=63  8*9=72  9*9=81
```



```cpp
for (int j = 1; j <= 9; j++) {
    for (int i = 1; i <= 3; i++) {
        cout << i << "*" << j << "=" << setw(2) << i * j << "  ";
    }
    cout << "\n";
}
cout << "----------------------\n";
for (int j = 1; j <= 9; j++) {
    for (int i = 4; i <= 6; i++) {
        cout << i << "*" << j << "=" << setw(2) << i * j << "  ";
    }
    cout << "\n";
}
cout << "----------------------\n";
for (int j = 1; j <= 9; j++) {
    for (int i = 7; i <= 9; i++) {
        cout << i << "*" << j << "=" << setw(2) << i * j << "  ";
    }
    cout << "\n";
}
```

多維陣列

二維陣列



	0	1	2
0	(0, 0)	(0, 1)	(0, 2)
1	(1, 0)	(1, 1)	(1, 2)

```
char a[2][3];
```

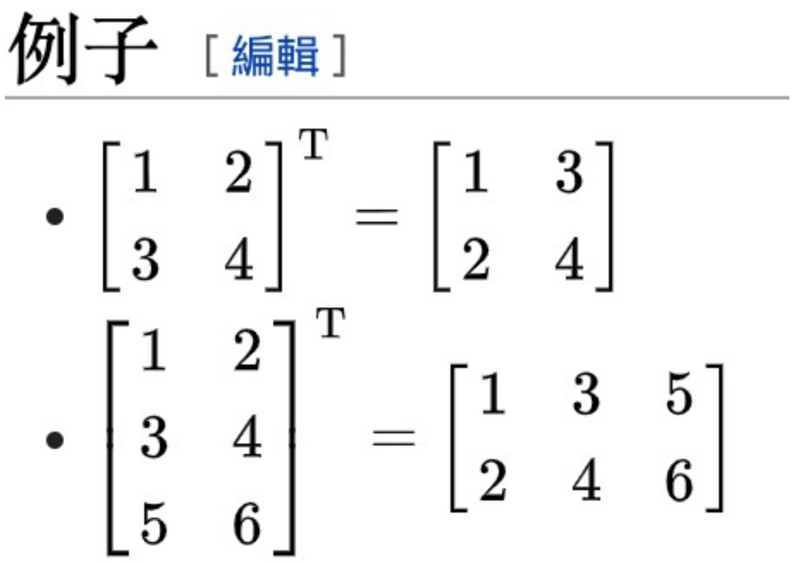
	0	1	2
0	A	B	C
1	D	E	F

```
a[0][2] = ?
```

In memory, row-major order

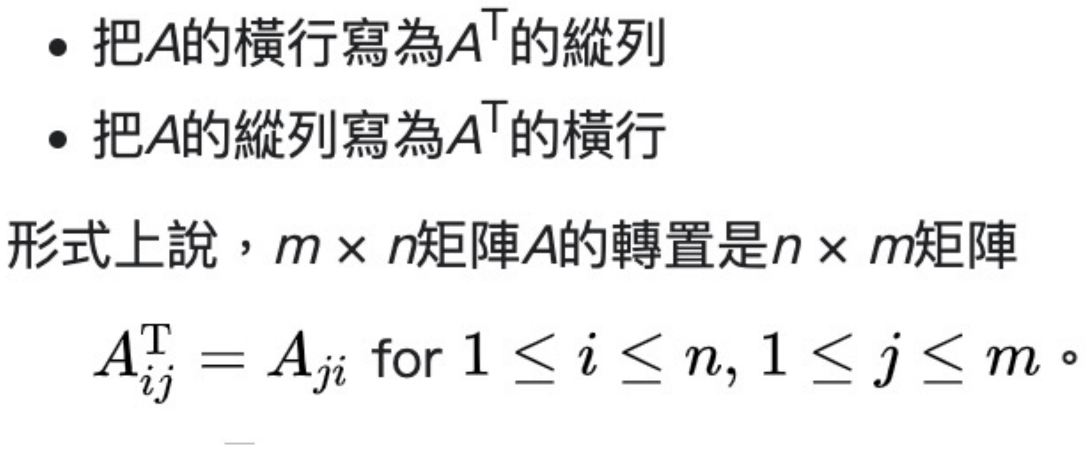
address	value	index
0x16fdff474	A	a[0][0]
0x16fdff475	B	a[0][1]
0x16fdff476	C	a[0][2]
0x16fdff477	D	a[1][0]
0x16fdff478	E	a[1][1]
0x16fdff479	F	a[1][2]

轉置矩陣 (transpose)



例子 [編輯]

- $\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}^{\mathrm{T}} = \begin{bmatrix}1 & 3\\2 & 4\end{bmatrix}$
- $\begin{bmatrix}1 & 2\\3 & 4\\5 & 6\end{bmatrix}^{\mathrm{T}} = \begin{bmatrix}1 & 3 & 5\\2 & 4 & 6\end{bmatrix}$



- 把A的橫行寫為A^{T}的縱列
- 把A的縱列寫為A^{T}的橫行

形式上說，$m \times n$矩陣A的轉置是$n \times m$矩陣

$$A^{\mathrm{T}}_{ij} = A_{ji} \text{ for } 1 \le i \le n, 1 \le j \le m$$ 。

參考資料：https://zh.wikipedia.org/wiki/%E8%BD%AC%E7%BD%AE%E7%9F%A9%E9%98%B5

【作業】a015: 矩陣的翻轉

注意：本題有多筆測資

【Input】

```
2 3
3 1 2
8 5 4
```

【Output】

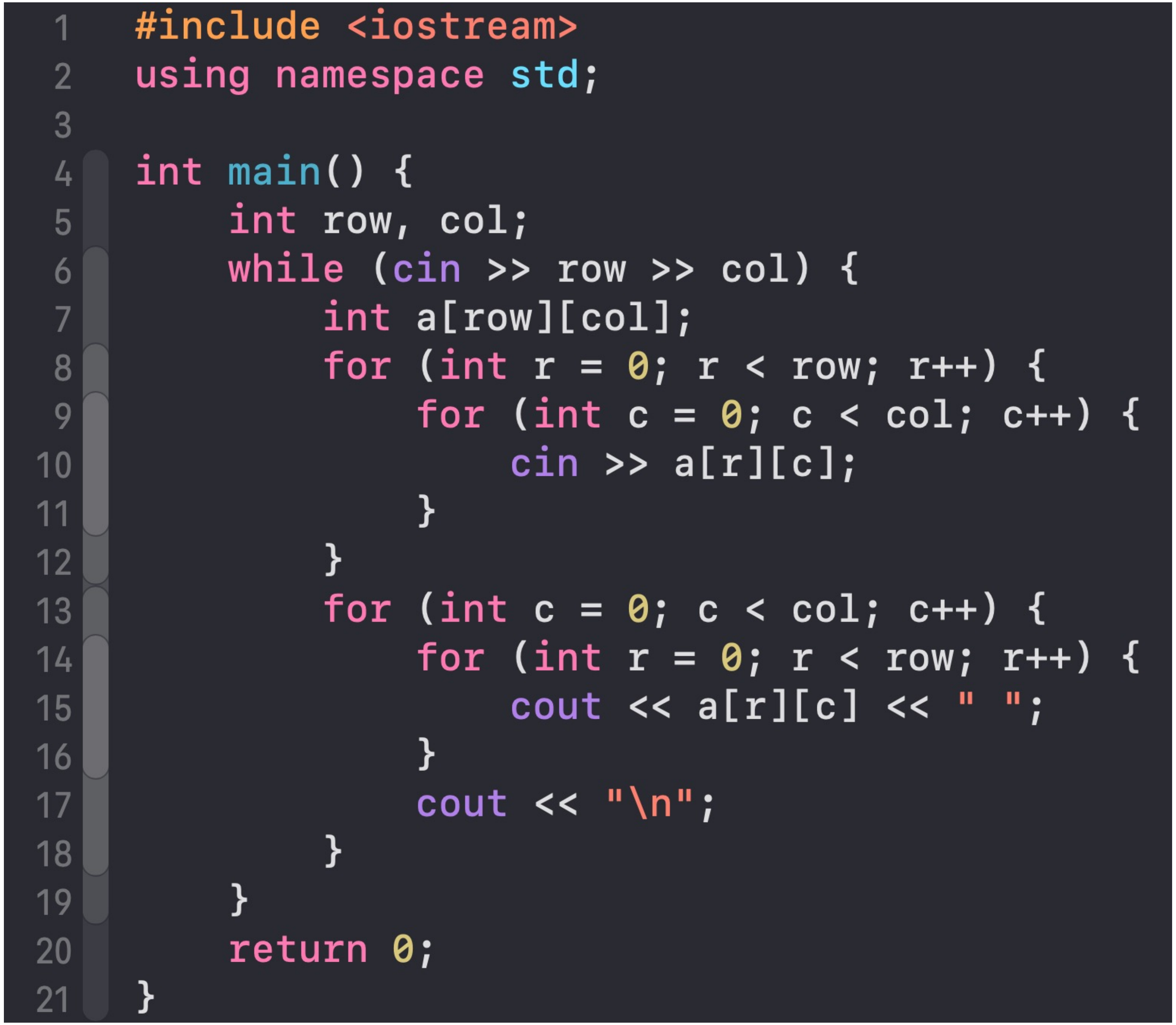
```
3 8
1 5
2 4
```

$$\begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & 3 & 1 & 2 \\ 1 & 8 & 5 & 4 \end{array} \xrightarrow{\text{transpose}} \begin{array}{c|cc} & 0 & 1 \\ \hline 0 & 3 & 8 \\ 1 & 1 & 5 \\ 3 & 2 & 4 \end{array}$$

Dimension: 2 x 3

Dimension: 3 x 2

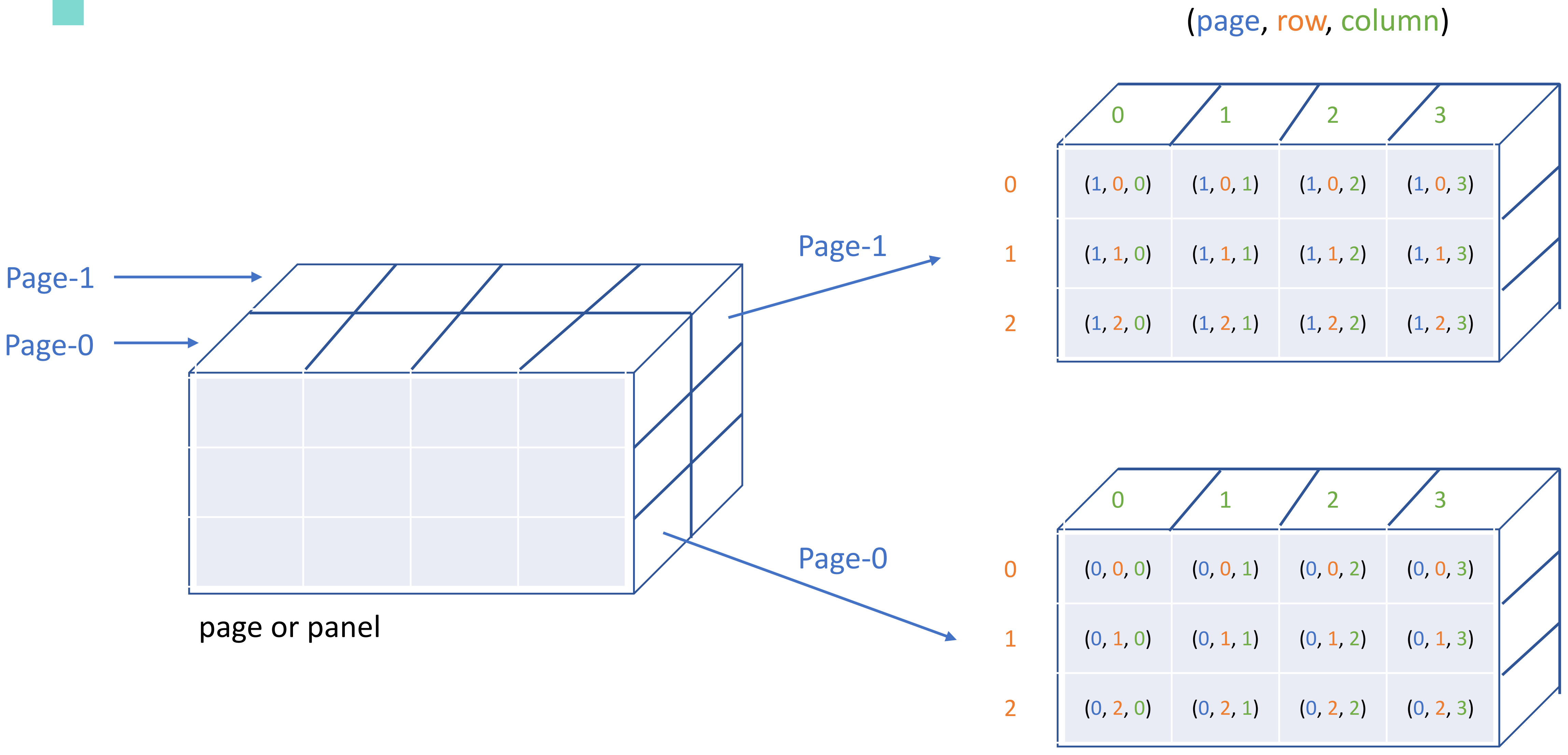
Example 6-1



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

int main() {
    int row, col;
    while (cin >> row >> col) {
        int a[row][col];
        for (int r = 0; r < row; r++) {
            for (int c = 0; c < col; c++) {
                cin >> a[r][c];
            }
        }
        for (int c = 0; c < col; c++) {
            for (int r = 0; r < row; r++) {
                cout << a[r][c] << " ";
            }
            cout << "\n";
        }
    }
    return 0;
}
```

三維陣列

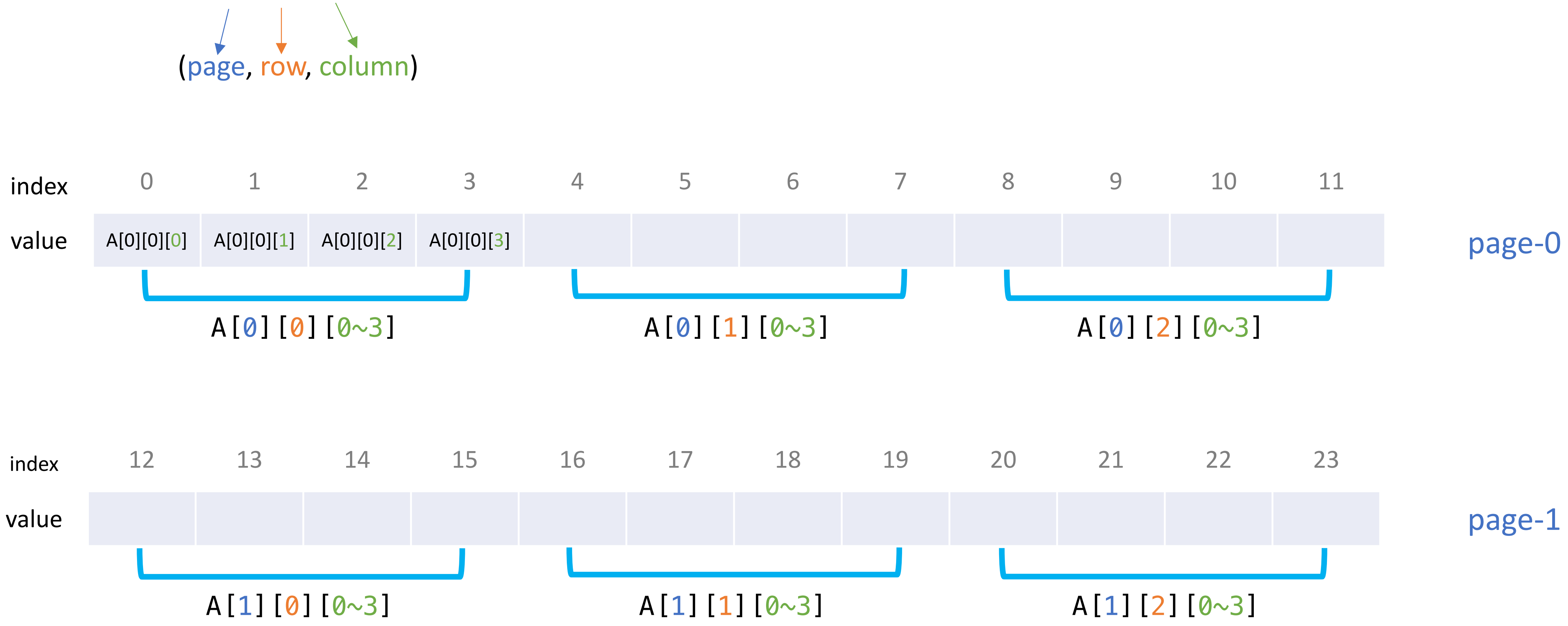


	0	1	2	3
0	(1, 0, 0)	(1, 0, 1)	(1, 0, 2)	(1, 0, 3)
1	(1, 1, 0)	(1, 1, 1)	(1, 1, 2)	(1, 1, 3)
2	(1, 2, 0)	(1, 2, 1)	(1, 2, 2)	(1, 2, 3)

	0	1	2	3
0	(0, 0, 0)	(0, 0, 1)	(0, 0, 2)	(0, 0, 3)
1	(0, 1, 0)	(0, 1, 1)	(0, 1, 2)	(0, 1, 3)
2	(0, 2, 0)	(0, 2, 1)	(0, 2, 2)	(0, 2, 3)

三維陣列

int A[2][3][4];



(page, row, column)

index	0	1	2	3	4	5	6	7	8	9	10	11
value	A[0][0][0]	A[0][0][1]	A[0][0][2]	A[0][0][3]								

A[0][0][0~3]　A[0][1][0~3]　A[0][2][0~3]

page-0

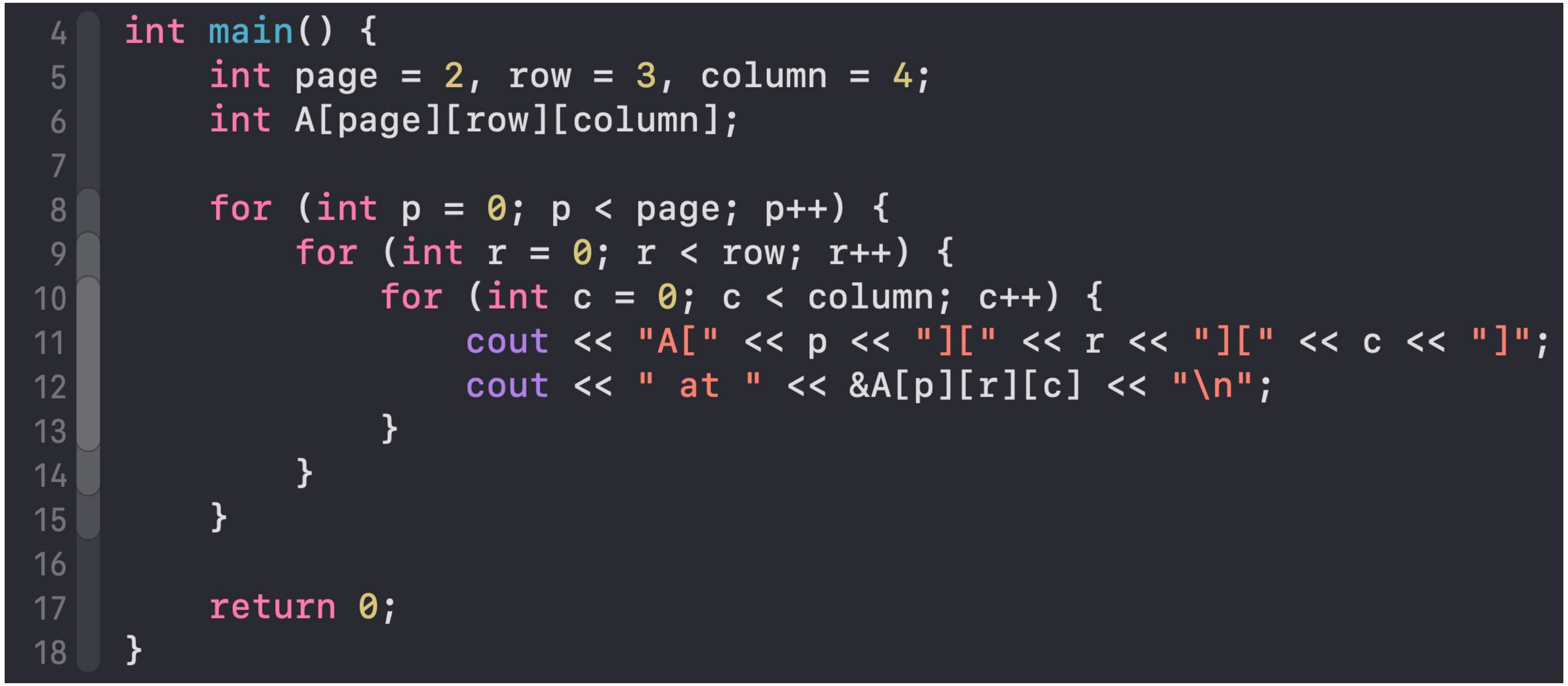
index	12	13	14	15	16	17	18	19	20	21	22	23
value												

A[1][0][0~3]　A[1][1][0~3]　A[1][2][0~3]

page-1

記憶體順序

Example 6-7

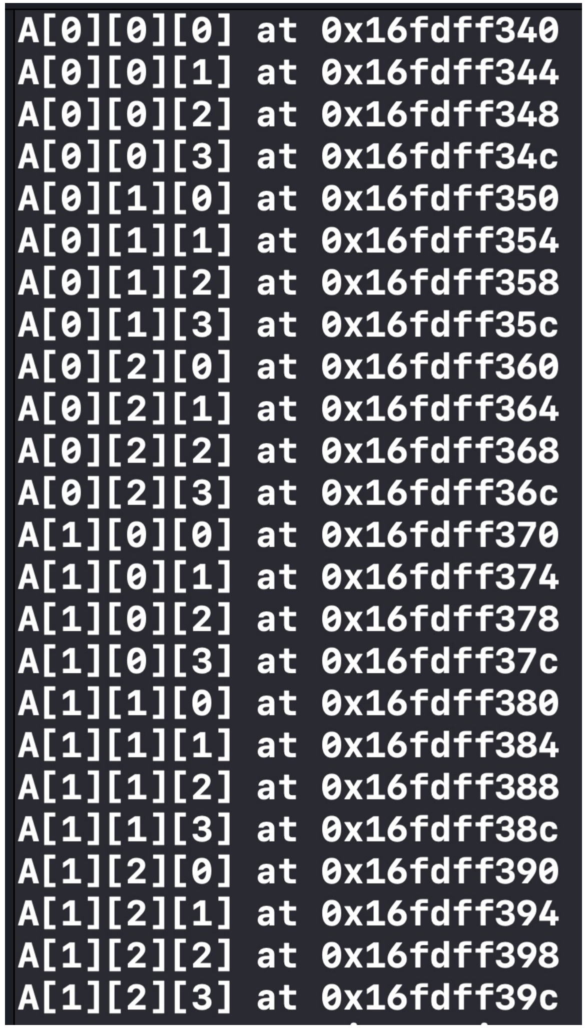


```cpp
int main() {
    int page = 2, row = 3, column = 4;
    int A[page][row][column];

    for (int p = 0; p < page; p++) {
        for (int r = 0; r < row; r++) {
            for (int c = 0; c < column; c++) {
                cout << "A[" << p << "][" << r << "][" << c << "]";
                cout << " at " << &A[p][r][c] << "\n";
            }
        }
    }

    return 0;
}
```

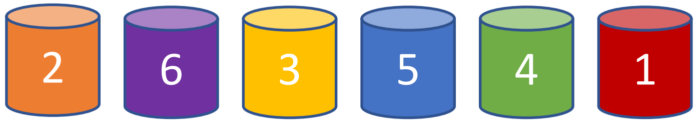
連續的記憶體位址



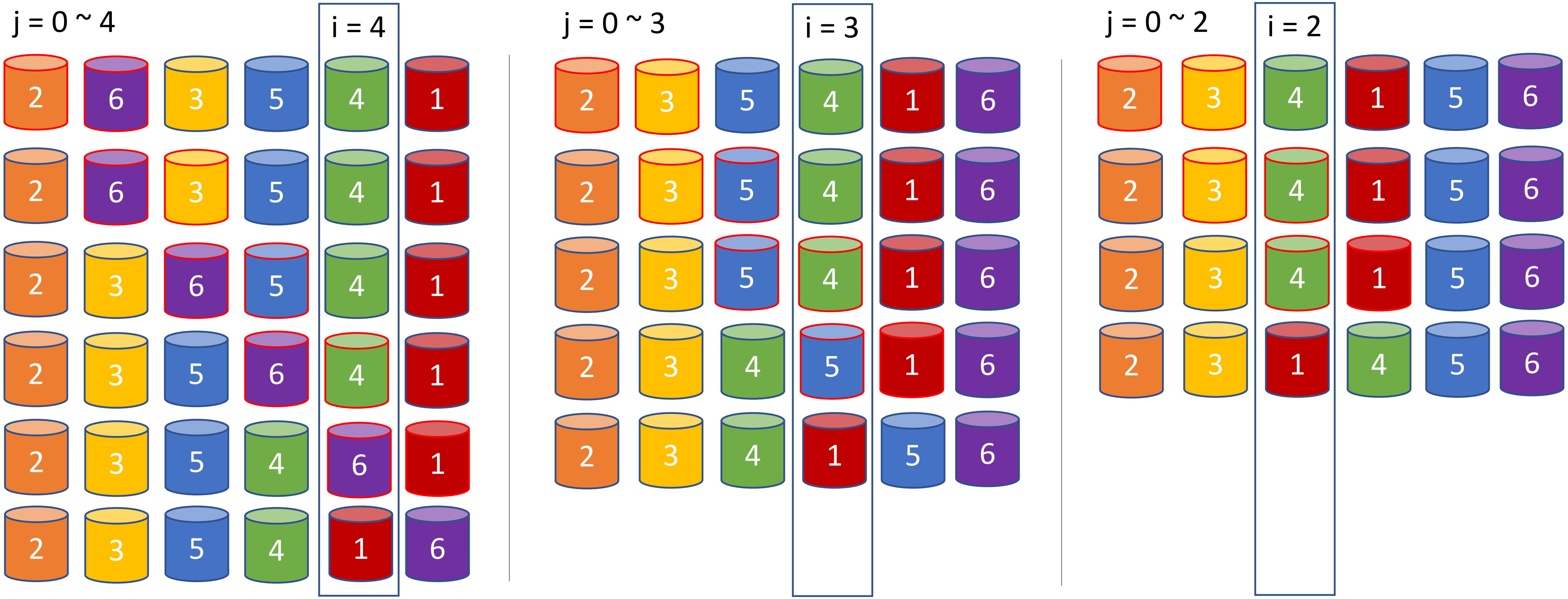
```
A[0][0][0] at 0x16fdff340
A[0][0][1] at 0x16fdff344
A[0][0][2] at 0x16fdff348
A[0][0][3] at 0x16fdff34c
A[0][1][0] at 0x16fdff350
A[0][1][1] at 0x16fdff354
A[0][1][2] at 0x16fdff358
A[0][1][3] at 0x16fdff35c
A[0][2][0] at 0x16fdff360
A[0][2][1] at 0x16fdff364
A[0][2][2] at 0x16fdff368
A[0][2][3] at 0x16fdff36c
A[1][0][0] at 0x16fdff370
A[1][0][1] at 0x16fdff374
A[1][0][2] at 0x16fdff378
A[1][0][3] at 0x16fdff37c
A[1][1][0] at 0x16fdff380
A[1][1][1] at 0x16fdff384
A[1][1][2] at 0x16fdff388
A[1][1][3] at 0x16fdff38c
A[1][2][0] at 0x16fdff390
A[1][2][1] at 0x16fdff394
A[1][2][2] at 0x16fdff398
A[1][2][3] at 0x16fdff39c
```

【範例】Bubble Sort 泡沫排序法

(動畫演示)



2 6 3 5 4 1



j = 0 ~ 4, i = 4

2 6 3 5 4 1
2 6 3 5 4 1
2 3 6 5 4 1
2 3 5 6 4 1
2 3 5 4 6 1
2 3 5 4 1 6

j = 0 ~ 3, i = 3

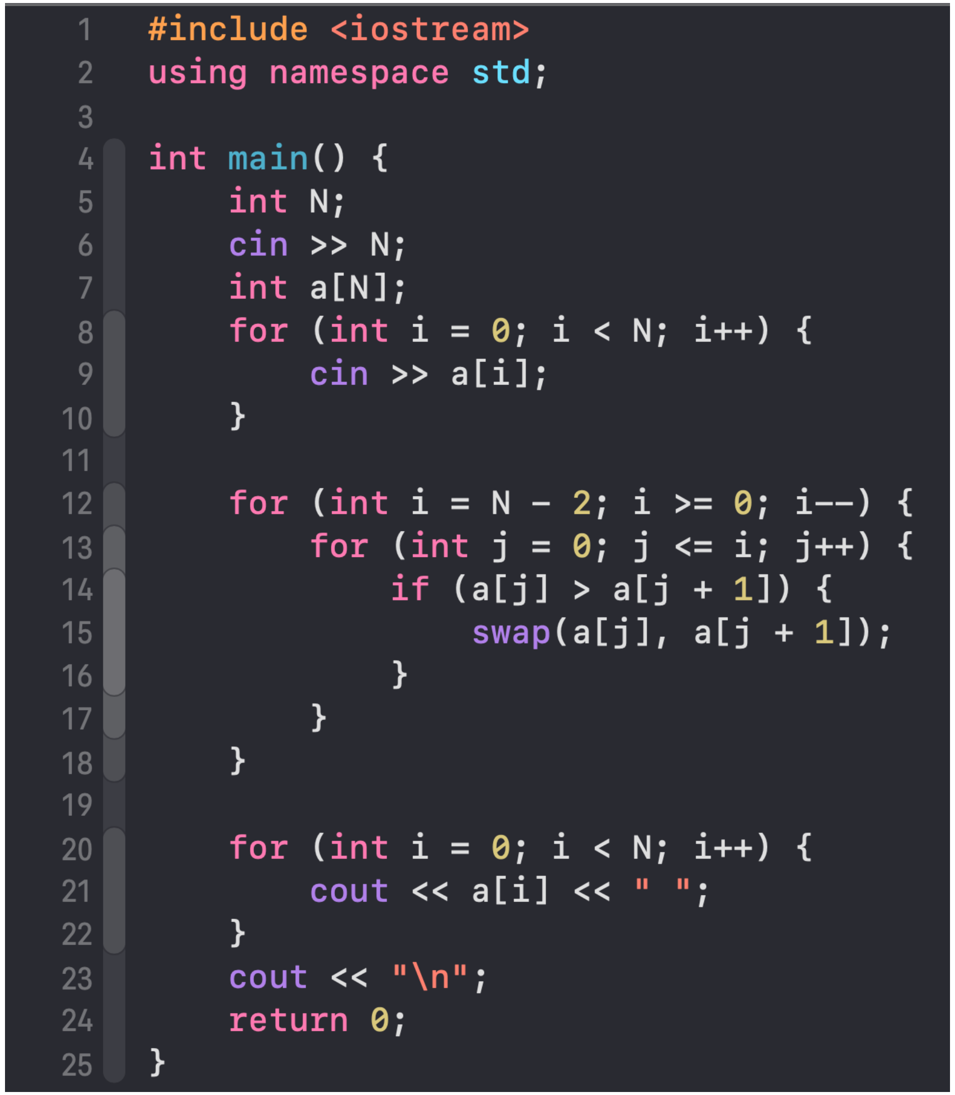
2 3 5 4 1 6
2 3 5 4 1 6
2 3 5 4 1 6
2 3 4 5 1 6
2 3 4 1 5 6

j = 0 ~ 2, i = 2

2 3 4 1 5 6
2 3 4 1 5 6
2 3 4 1 5 6
2 3 1 4 5 6

【範例】Bubble Sort 泡沫排序法

Example 6-5



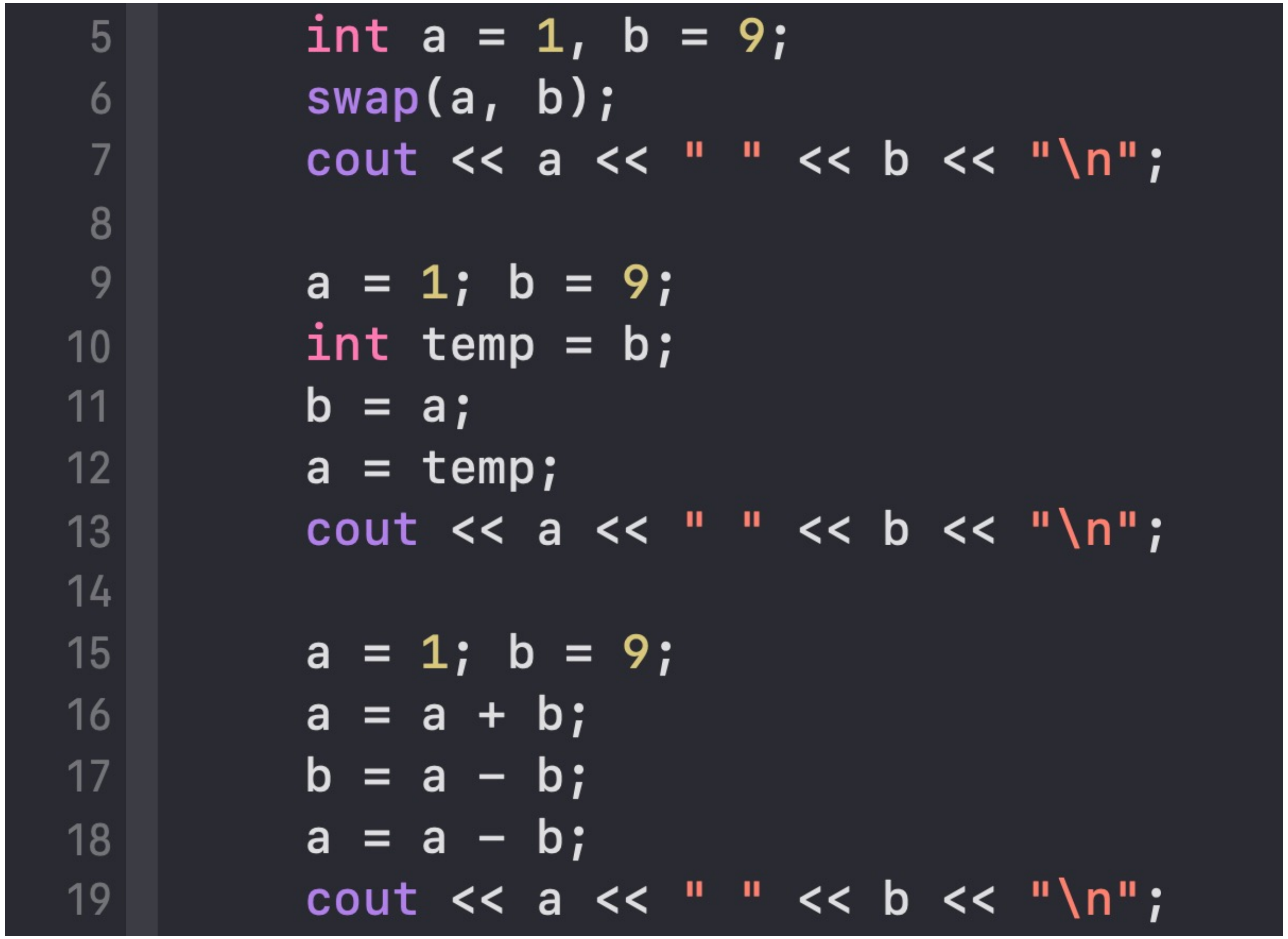
```cpp
#include <iostream>
using namespace std;

int main() {
    int N;
    cin >> N;
    int a[N];
    for (int i = 0; i < N; i++) {
        cin >> a[i];
    }

    for (int i = N - 2; i >= 0; i--) {
        for (int j = 0; j <= i; j++) {
            if (a[j] > a[j + 1]) {
                swap(a[j], a[j + 1]);
            }
        }
    }

    for (int i = 0; i < N; i++) {
        cout << a[i] << " ";
    }
    cout << "\n";
    return 0;
}
```

【補充】三種交換 a, b 的方法



```cpp
    int a = 1, b = 9;
    swap(a, b);
    cout << a << " " << b << "\n";

    a = 1; b = 9;
    int temp = b;
    b = a;
    a = temp;
    cout << a << " " << b << "\n";

    a = 1; b = 9;
    a = a + b;
    b = a - b;
    a = a - b;
    cout << a << " " << b << "\n";
```

【範例】質數篩法

Sieve of Eratosthenes 質數篩法 [Wikipedia]

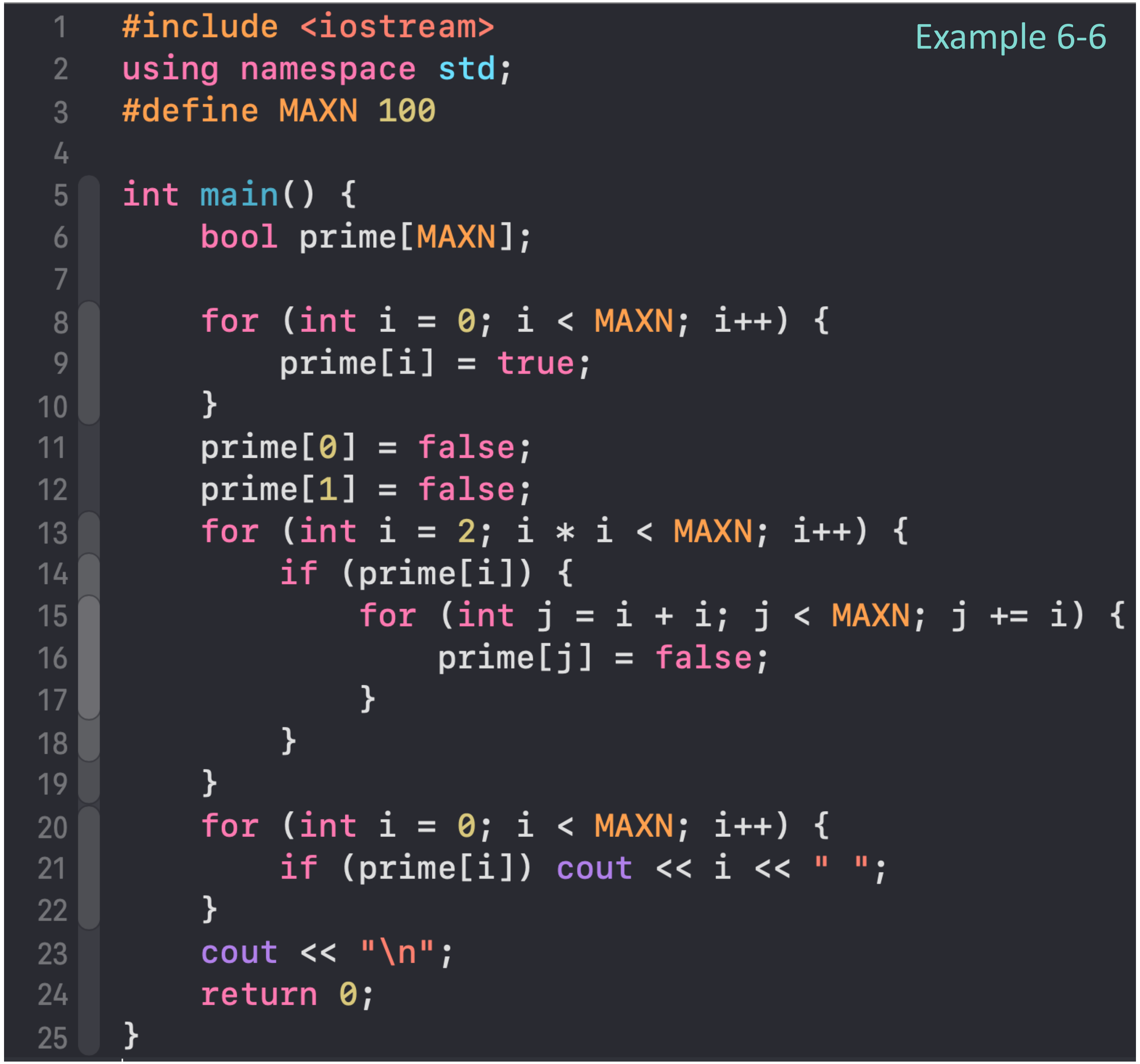
index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
prime	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T
	F	F	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T

… …

… …

… …

【範例】質數篩法



Example 6-6

```cpp
#include <iostream>
using namespace std;
#define MAXN 100

int main() {
    bool prime[MAXN];

    for (int i = 0; i < MAXN; i++) {
        prime[i] = true;
    }
    prime[0] = false;
    prime[1] = false;
    for (int i = 2; i * i < MAXN; i++) {
        if (prime[i]) {
            for (int j = i + i; j < MAXN; j += i) {
                prime[j] = false;
            }
        }
    }
    for (int i = 0; i < MAXN; i++) {
        if (prime[i]) cout << i << " ";
    }
    cout << "\n";
    return 0;
}
```