2017

程式設計

加強班

程式設計與實習(一)

BY 孫茂勛

Email:JOHN85051232@GMAIL.COM

Debug工具

- 來不及做這個的講義了囧
- I/O、中斷點、逐步執行(F10/F11)

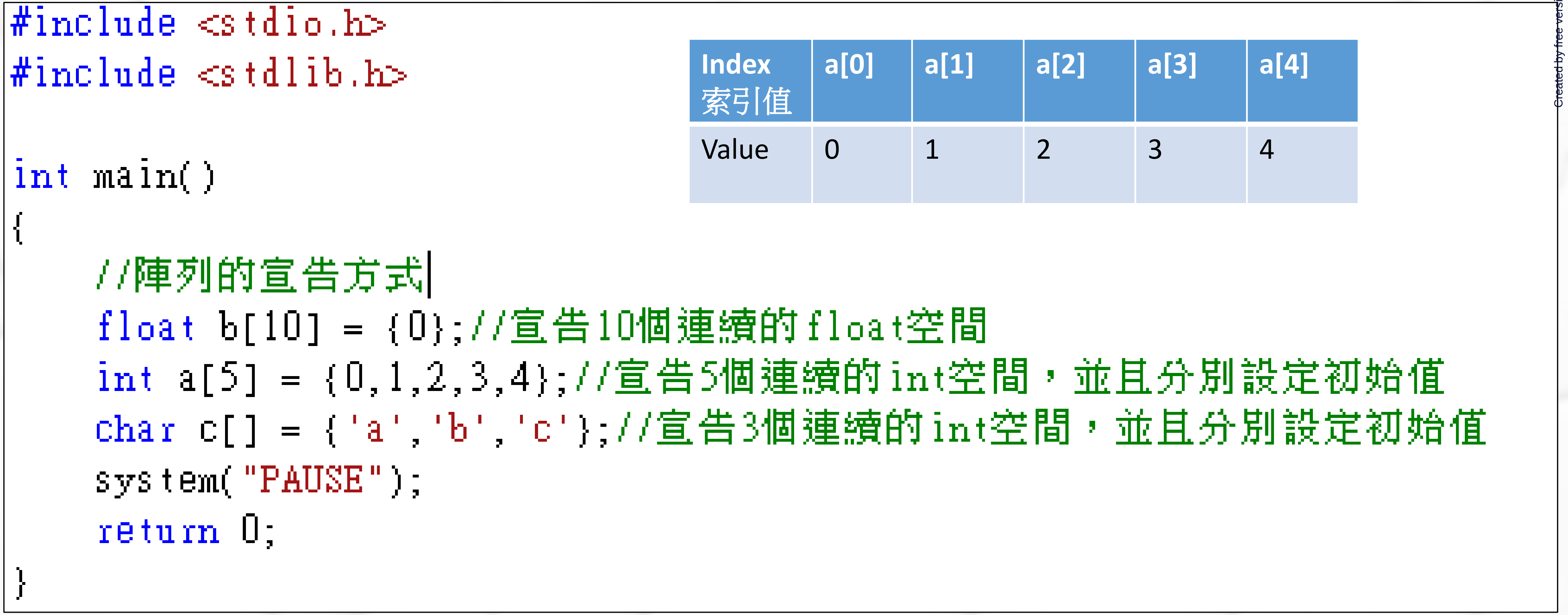
複習Array

陣列(Array)

- 一次取得一串連續的記憶體空間。
- C的陣列元素必須是相同的資料型態。
- 宣告方式：資料型態 陣列名稱[陣列大小]
- 陣列大小不要用變數(CH.6-12)

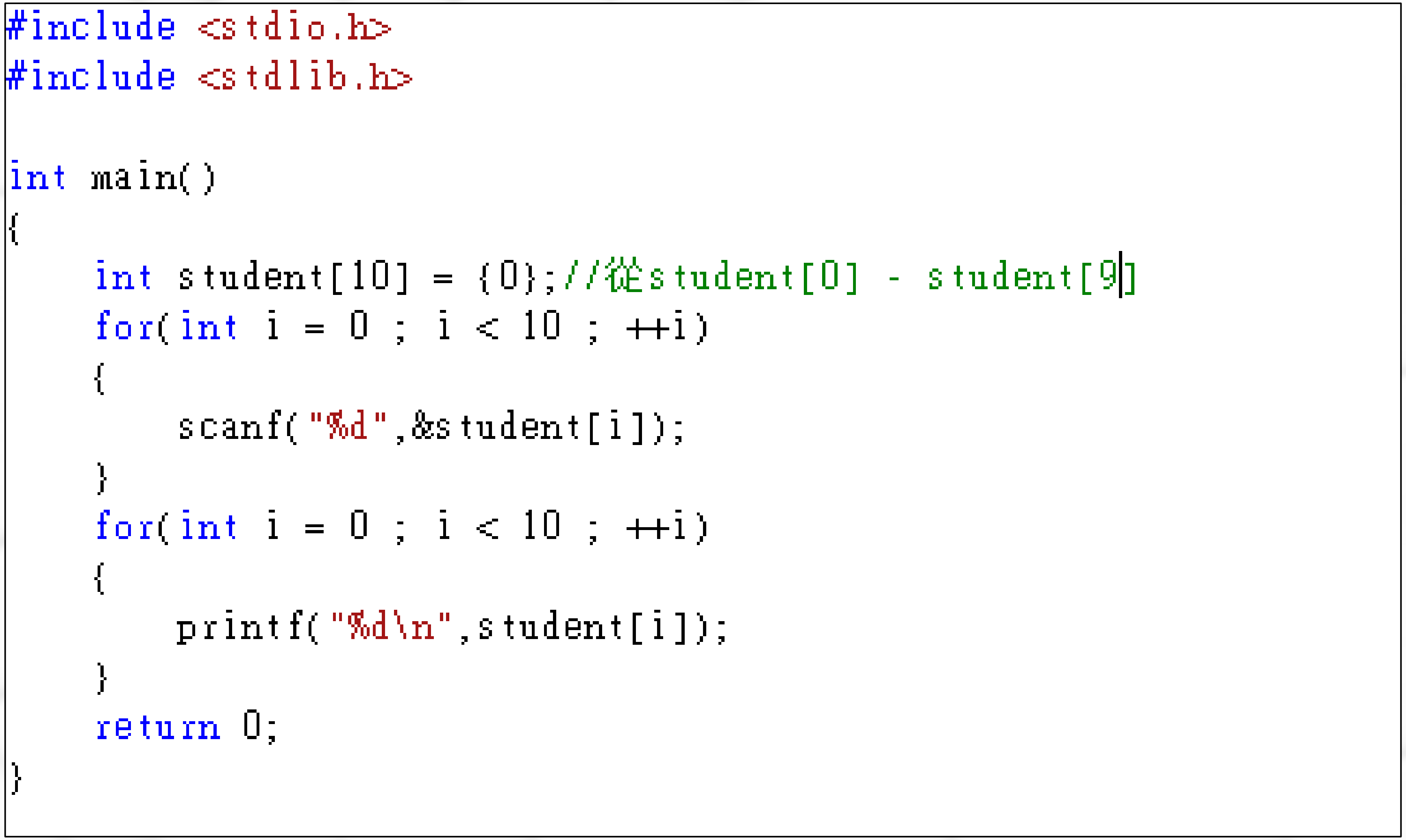
複習Array

Index 索引值	a[0]	a[1]	a[2]	a[3]	a[4]
Value	0	1	2	3	4

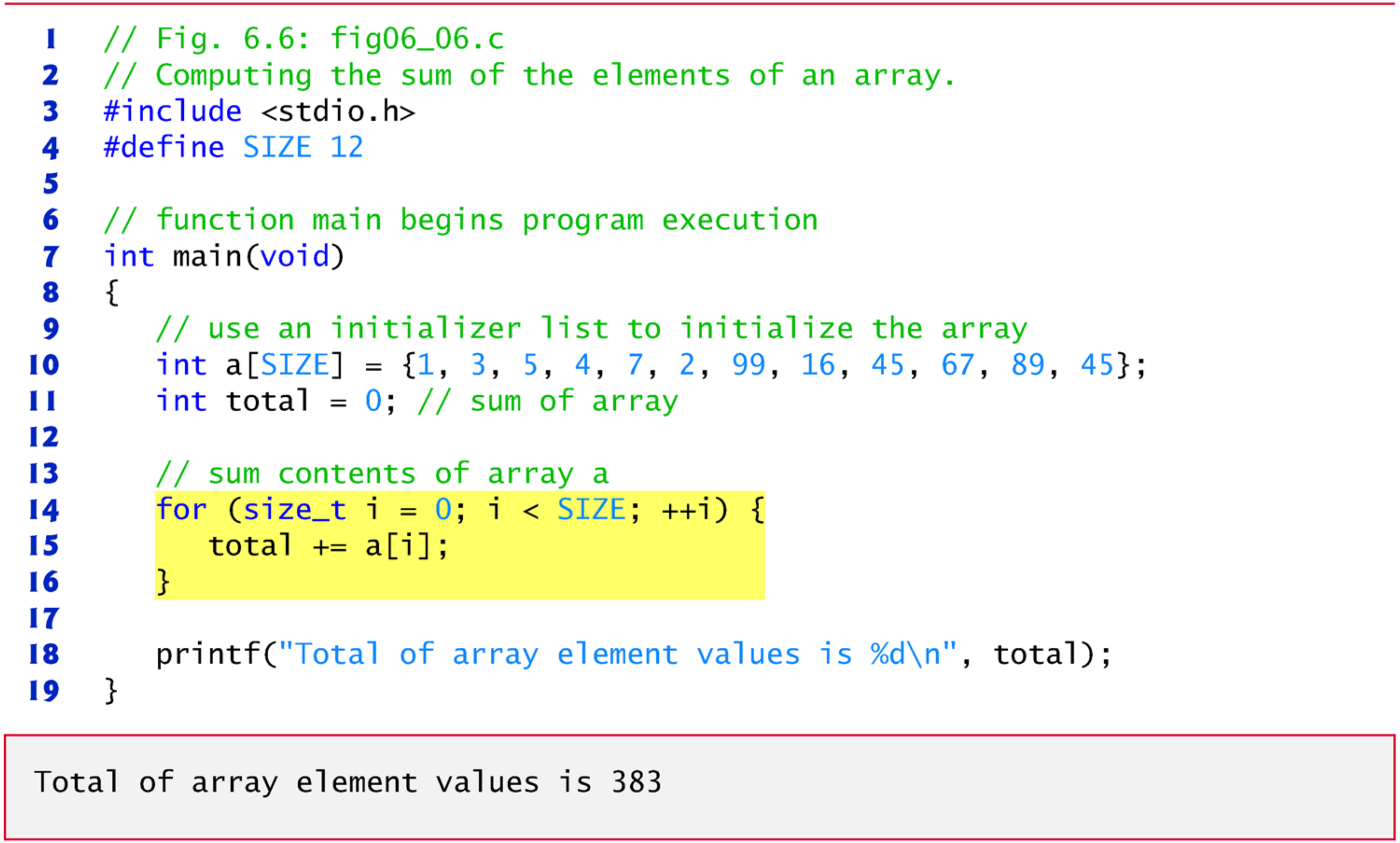


```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    //陣列的宣告方式
    float b[10] = {0};//宣告10個連續的float空間
    int a[5] = {0,1,2,3,4};//宣告5個連續的int空間，並且分別設定初始值
    char c[] = {'a','b','c'};//宣告3個連續的int空間，並且分別設定初始值
    system("PAUSE");
    return 0;
}
```

複習Array



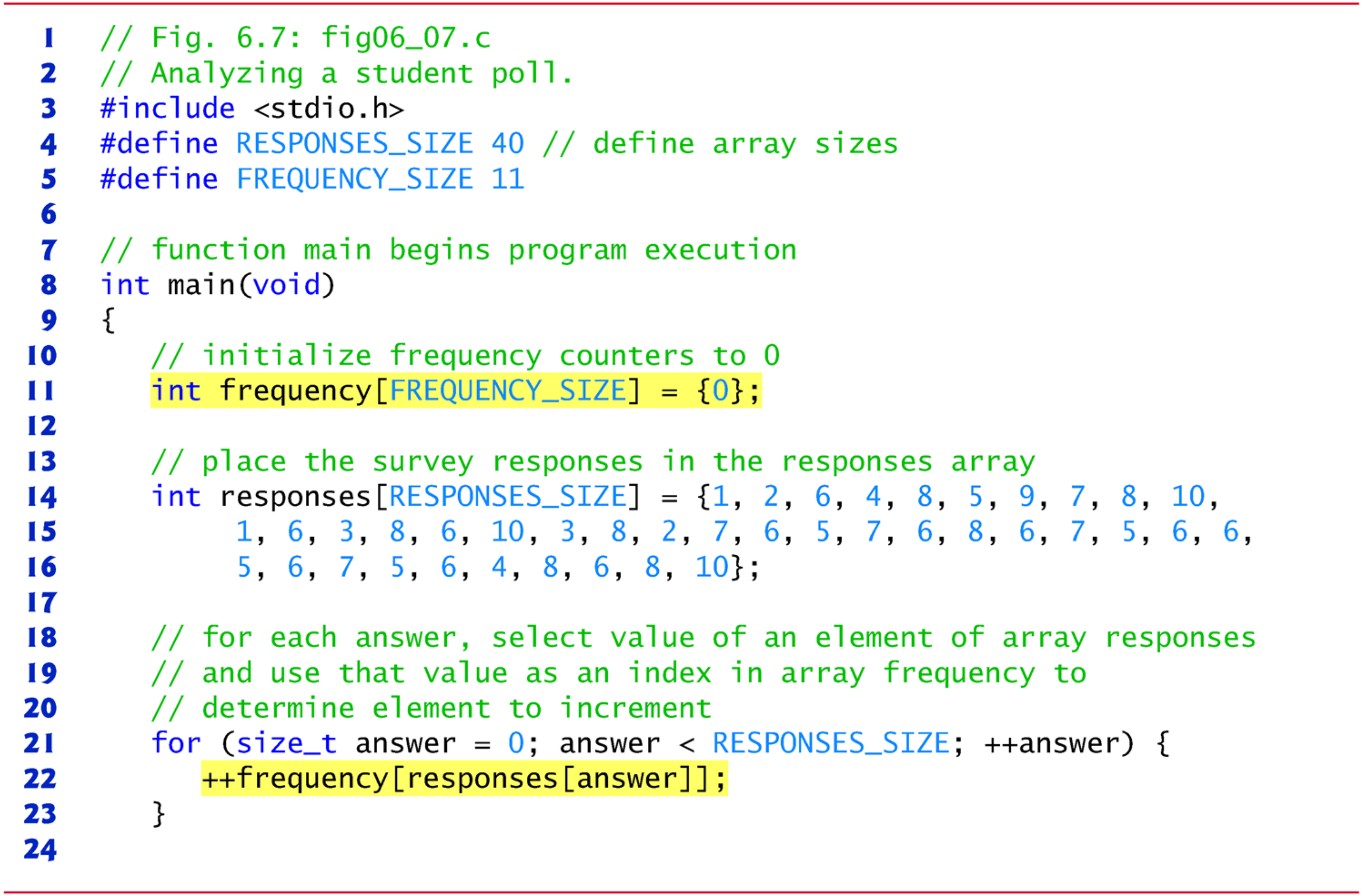
```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int student[10] = {0};//從student[0] - student[9]
    for(int i = 0 ; i < 10 ; ++i)
    {
        scanf("%d",&student[i]);
    }
    for(int i = 0 ; i < 10 ; ++i)
    {
        printf("%d\n",student[i]);
    }
    return 0;
}
```



```c
// Fig. 6.6: fig06_06.c
// Computing the sum of the elements of an array.
#include <stdio.h>
#define SIZE 12

// function main begins program execution
int main(void)
{
   // use an initializer list to initialize the array
   int a[SIZE] = {1, 3, 5, 4, 7, 2, 99, 16, 45, 67, 89, 45};
   int total = 0; // sum of array

   // sum contents of array a
   for (size_t i = 0; i < SIZE; ++i) {
      total += a[i];
   }

   printf("Total of array element values is %d\n", total);
}
```

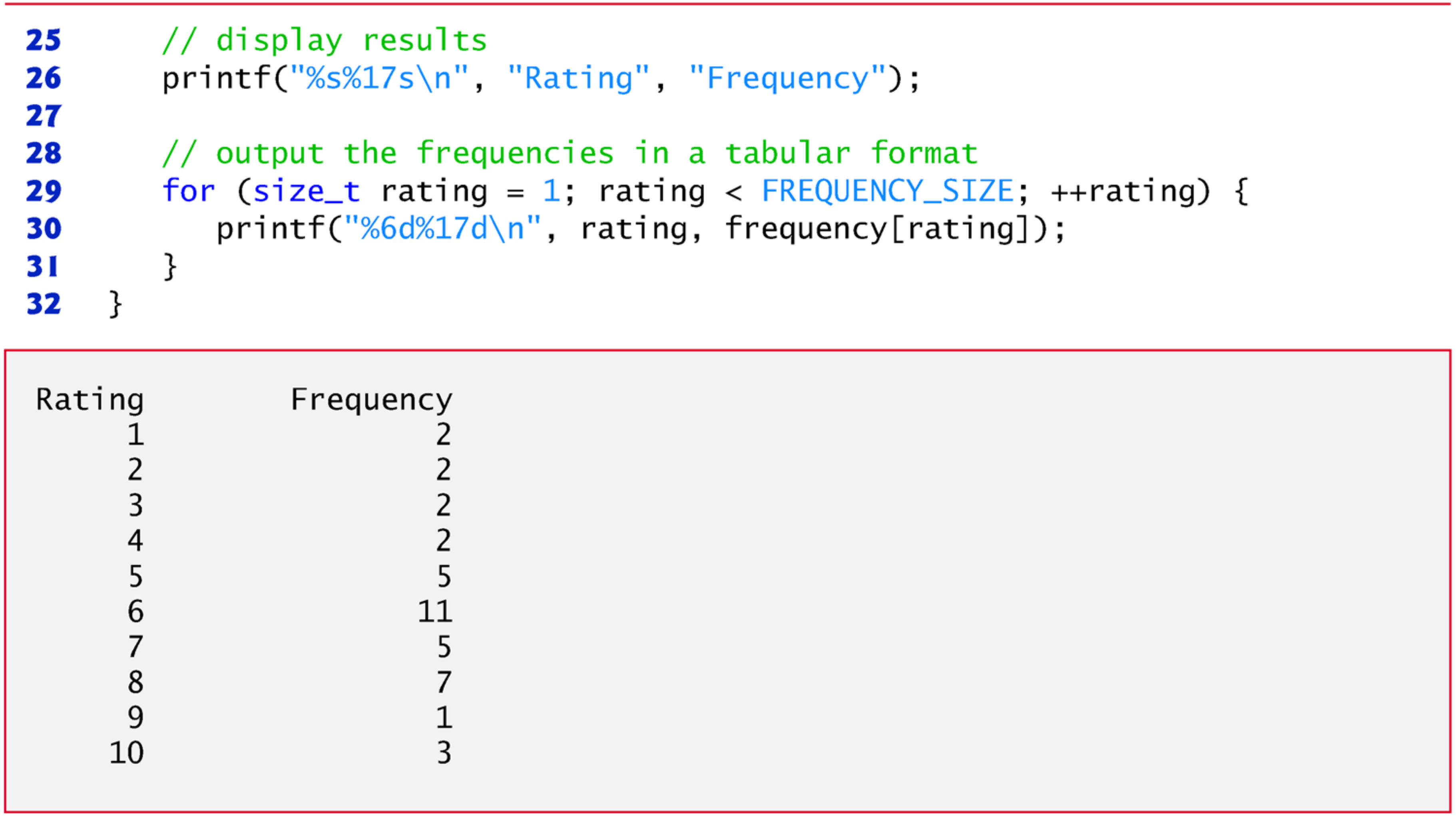
```
Total of array element values is 383
```

Fig. 6.6 | Computing the sum of the elements of an array.



```c
// Fig. 6.7: fig06_07.c
// Analyzing a student poll.
#include <stdio.h>
#define RESPONSES_SIZE 40 // define array sizes
#define FREQUENCY_SIZE 11

// function main begins program execution
int main(void)
{
   // initialize frequency counters to 0
   int frequency[FREQUENCY_SIZE] = {0};

   // place the survey responses in the responses array
   int responses[RESPONSES_SIZE] = {1, 2, 6, 4, 8, 5, 9, 7, 8, 10,
        1, 6, 3, 8, 6, 10, 3, 8, 2, 7, 6, 5, 7, 6, 8, 6, 7, 5, 6, 6,
        5, 6, 7, 5, 6, 4, 8, 6, 8, 10};

   // for each answer, select value of an element of array responses
   // and use that value as an index in array frequency to
   // determine element to increment
   for (size_t answer = 0; answer < RESPONSES_SIZE; ++answer) {
      ++frequency[responses[answer]];
   }

```

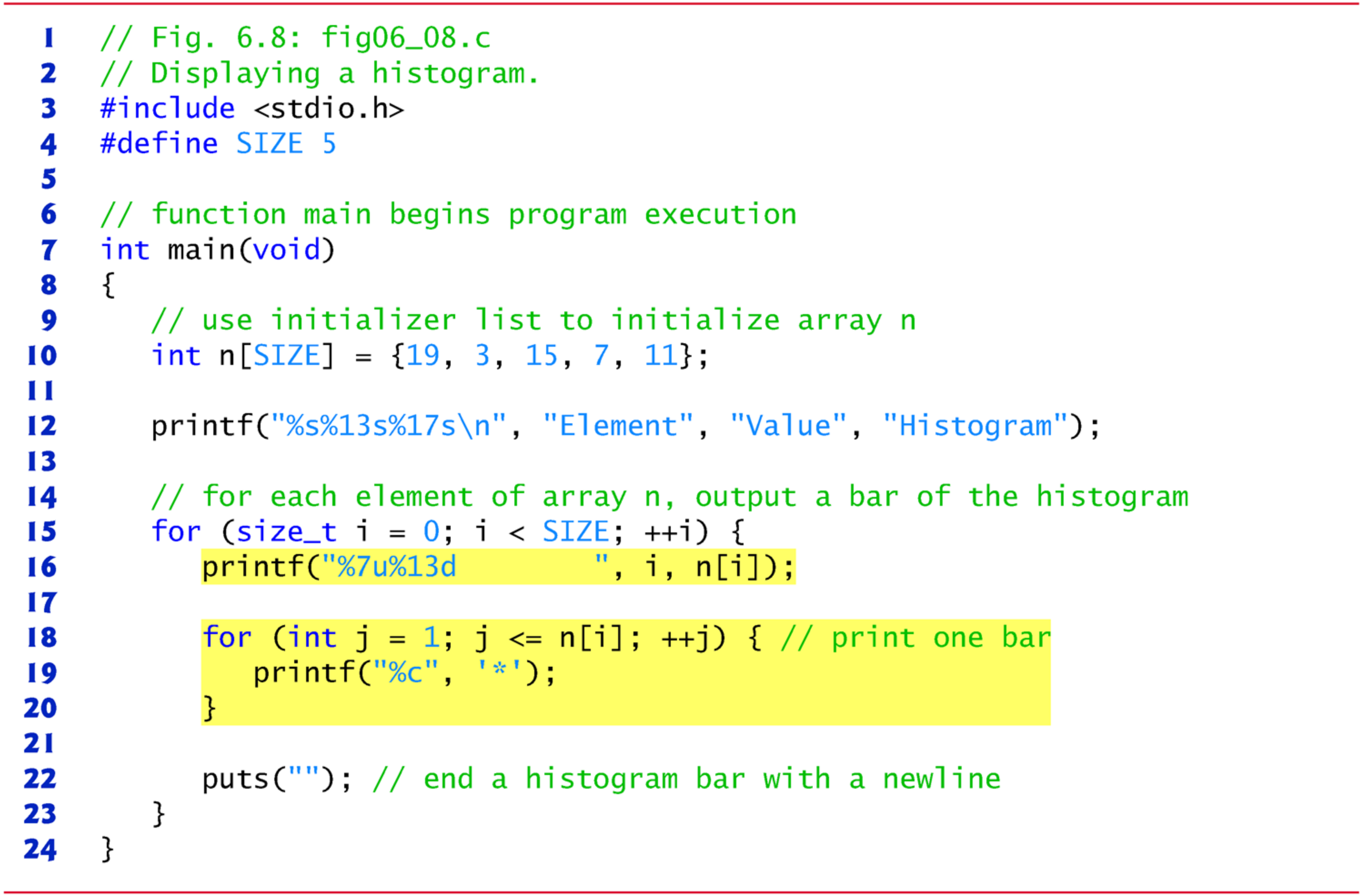
Fig. 6.7 | Analyzing a student poll. (Part 1 of 2.)



```c
   // display results
   printf("%s%17s\n", "Rating", "Frequency");

   // output the frequencies in a tabular format
   for (size_t rating = 1; rating < FREQUENCY_SIZE; ++rating) {
      printf("%6d%17d\n", rating, frequency[rating]);
   }
}
```

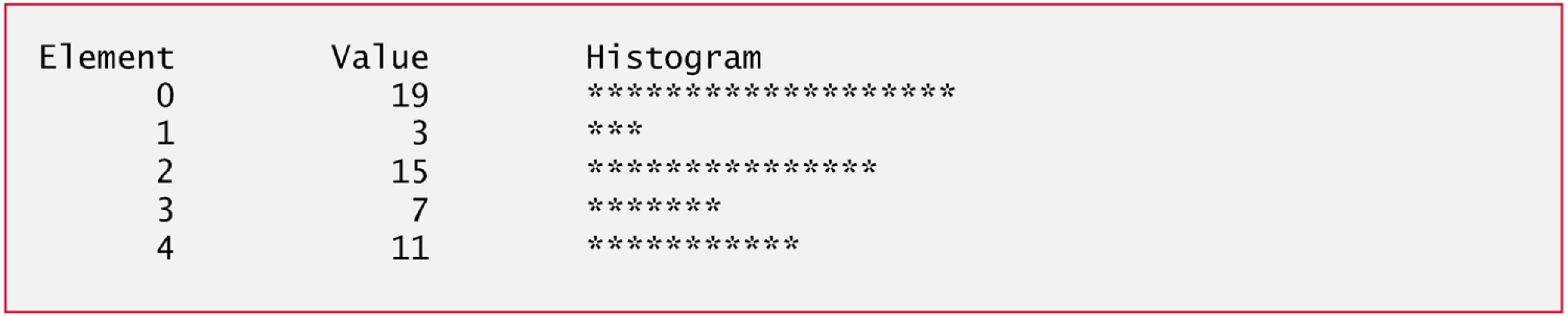
```
Rating        Frequency
     1                2
     2                2
     3                2
     4                2
     5                5
     6               11
     7                5
     8                7
     9                1
    10                3
```

Fig. 6.7 | Analyzing a student poll. (Part 2 of 2.)



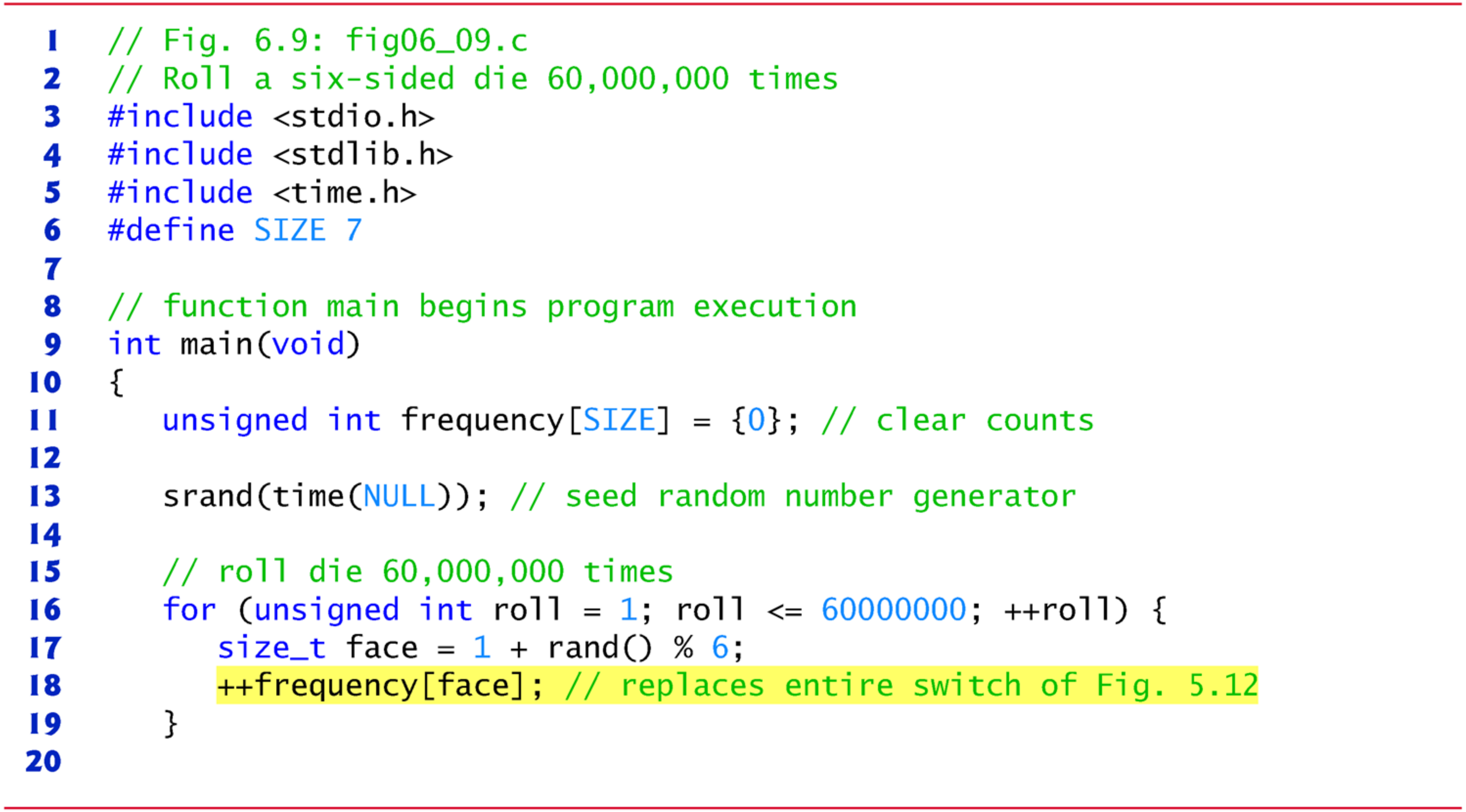
```c
// Fig. 6.8: fig06_08.c
// Displaying a histogram.
#include <stdio.h>
#define SIZE 5

// function main begins program execution
int main(void)
{
   // use initializer list to initialize array n
   int n[SIZE] = {19, 3, 15, 7, 11};

   printf("%s%13s%17s\n", "Element", "Value", "Histogram");

   // for each element of array n, output a bar of the histogram
   for (size_t i = 0; i < SIZE; ++i) {
      printf("%7u%13d        ", i, n[i]);

      for (int j = 1; j <= n[i]; ++j) { // print one bar
         printf("%c", '*');
      }

      puts(""); // end a histogram bar with a newline
   }
}
```

Fig. 6.8 | Displaying a histogram. (Part 1 of 2.)



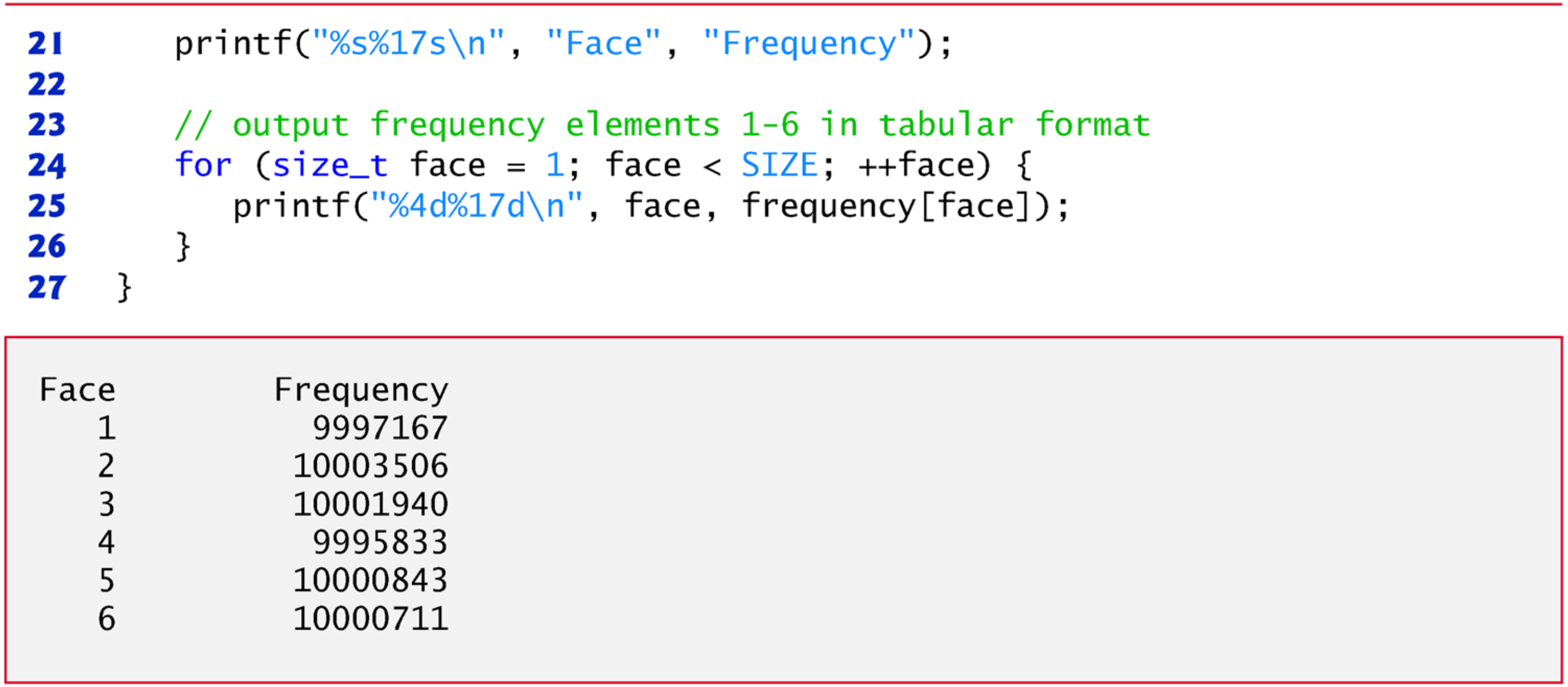
```
Element        Value        Histogram
      0           19        *******************
      1            3        ***
      2           15        ***************
      3            7        *******
      4           11        ***********
```

Fig. 6.8 | Displaying a histogram. (Part 2 of 2.)



```c
// Fig. 6.9: fig06_09.c
// Roll a six-sided die 60,000,000 times
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define SIZE 7

// function main begins program execution
int main(void)
{
   unsigned int frequency[SIZE] = {0}; // clear counts

   srand(time(NULL)); // seed random number generator

   // roll die 60,000,000 times
   for (unsigned int roll = 1; roll <= 60000000; ++roll) {
      size_t face = 1 + rand() % 6;
      ++frequency[face]; // replaces entire switch of Fig. 5.12
   }

```

Fig. 6.9 | Roll a six-sided die 60,000,000 times. (Part 1 of 2.)



```c
   printf("%s%17s\n", "Face", "Frequency");

   // output frequency elements 1-6 in tabular format
   for (size_t face = 1; face < SIZE; ++face) {
      printf("%4d%17d\n", face, frequency[face]);
   }
}
```

```
Face        Frequency
   1          9997167
   2         10003506
   3         10001940
   4          9995833
   5         10000843
   6         10000711
```

Fig. 6.9 | Roll a six-sided die 60,000,000 times. (Part 2 of 2.)

Binary Search

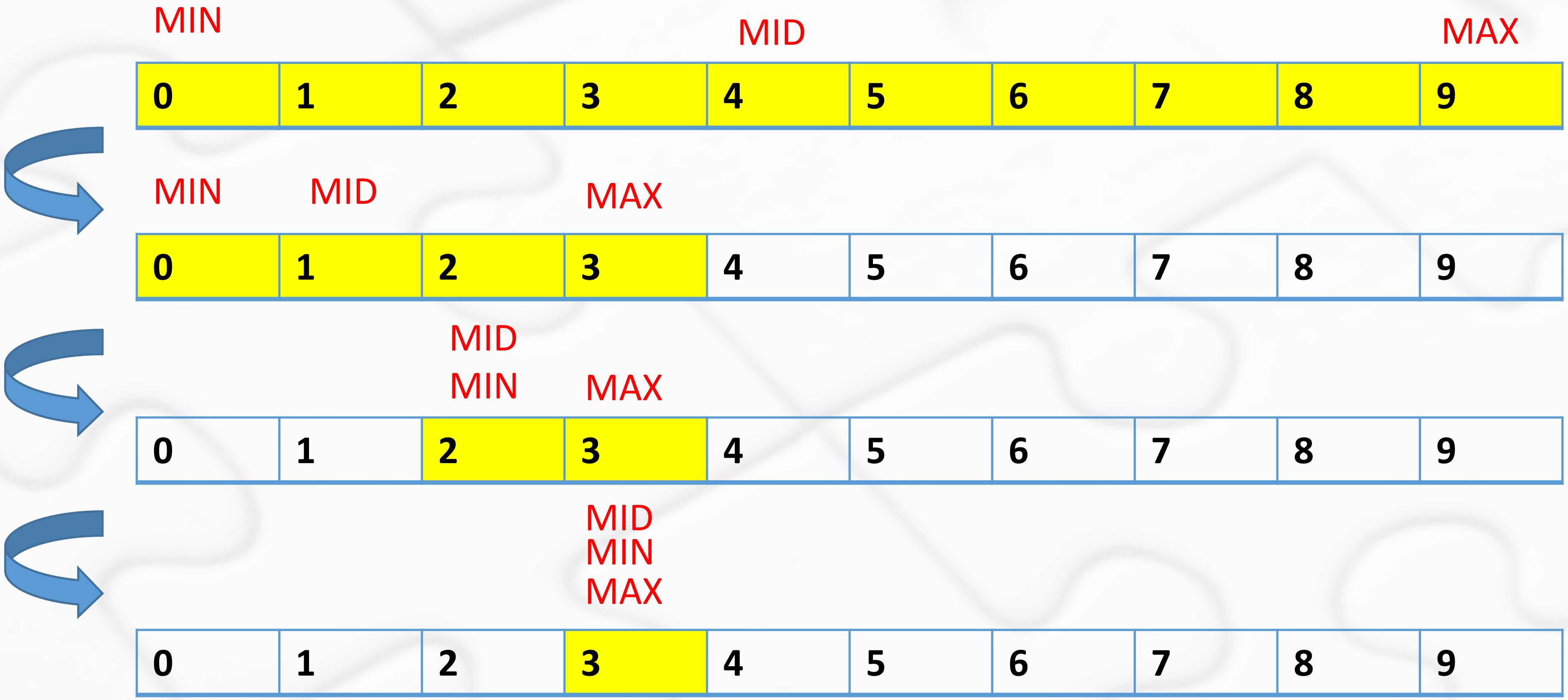
- 有沒有比之前更快的搜尋法?

Binary Search(二分搜尋法)

- 資料需要先排序好。
- 資料具有最大值、最小值、中間值三個比較點
  - 若搜尋的資料>中間值，則更新最小值
  - 若搜尋的資料<中間值，則更新最大值
  - 若搜尋的資料=中間值，就找到了
  - 若資料的最小值>最大值，則代表找不到資料

Binary Search

- Q：搜尋資料(0 1 2 3 4 5 6 7 8 9)中是否有 3 ？



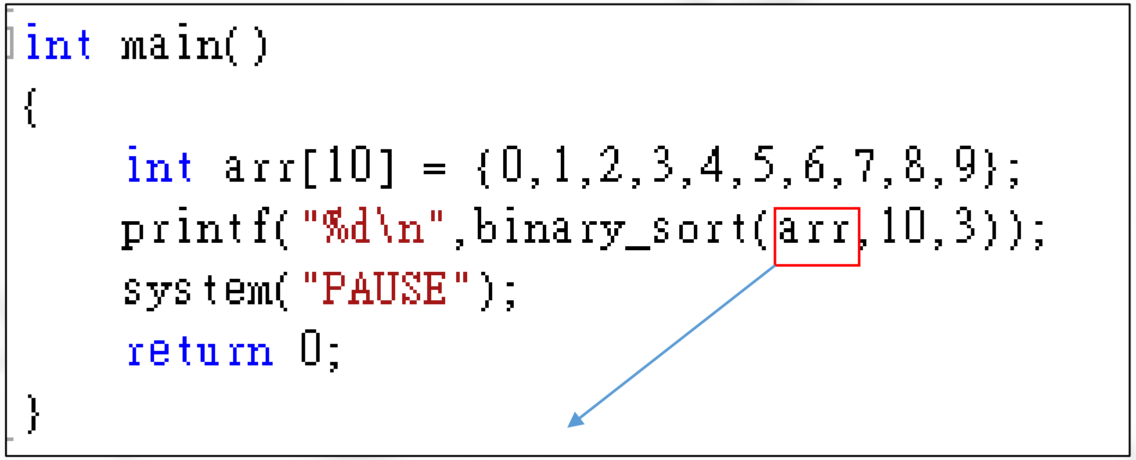
MIN				MID					MAX
0	1	2	3	4	5	6	7	8	9

MIN	MID		MAX						
0	1	2	3	4	5	6	7	8	9

		MID MIN	MAX						
0	1	2	3	4	5	6	7	8	9

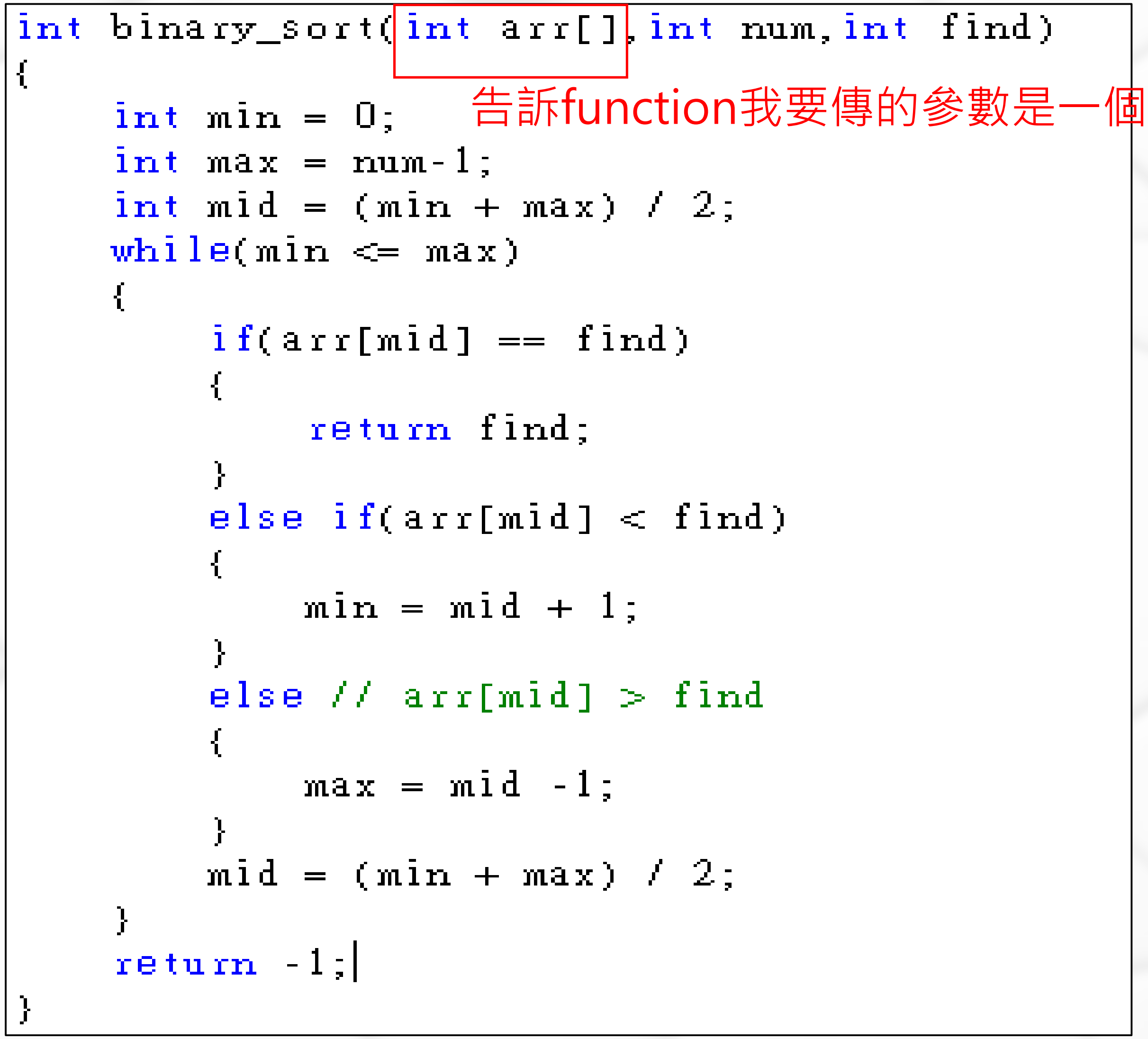
			MID MIN MAX						
0	1	2	3	4	5	6	7	8	9

Binary Search



```
int main()
{
    int arr[10] = {0,1,2,3,4,5,6,7,8,9};
    printf("%d\n",binary_sort(arr,10,3));
    system("PAUSE");
    return 0;
}
```

一維陣列的參數傳遞需要告知起始位置
(arr代表arr[0]的記憶體位址)



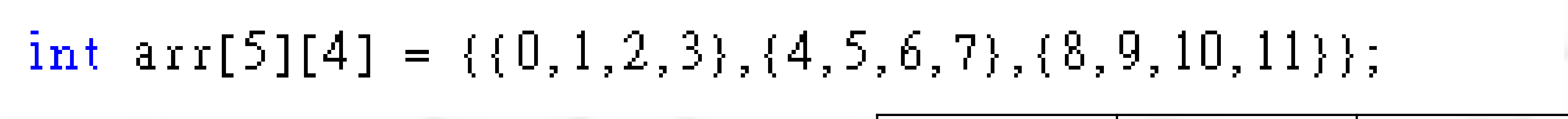
```
int binary_sort(int arr[],int num,int find)
{
    int min = 0;
    int max = num-1;
    int mid = (min + max) / 2;
    while(min <= max)
    {
        if(arr[mid] == find)
        {
            return find;
        }
        else if(arr[mid] < find)
        {
            min = mid + 1;
        }
        else // arr[mid] > find
        {
            max = mid -1;
        }
        mid = (min + max) / 2;
    }
    return -1;
}
```

告訴function我要傳的參數是一個陣列

Array

二維陣列

- 有點像矩陣
- 資料型態 陣列名稱[陣列大小] [陣列大小]



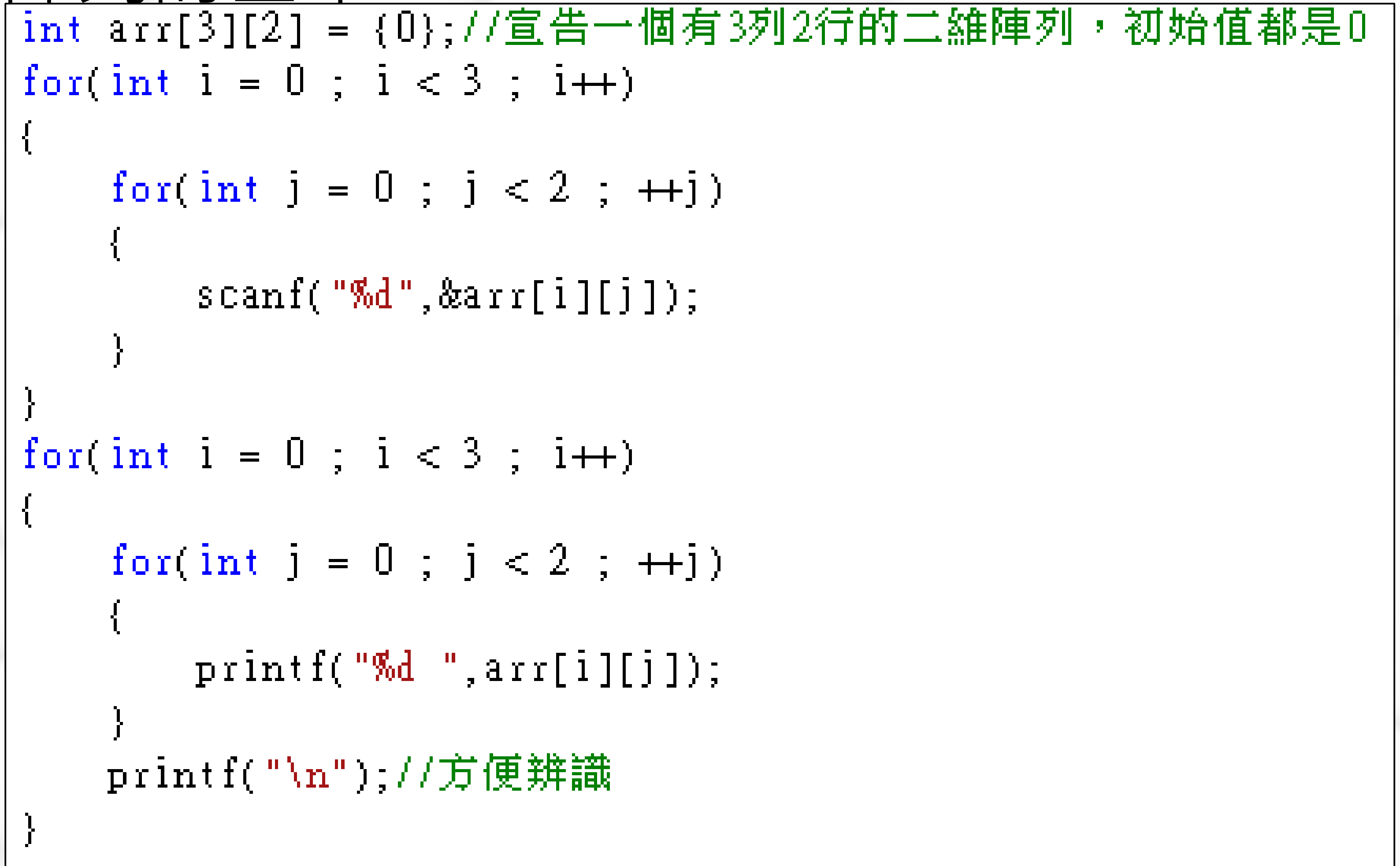
```
int arr[5][4] = {{0,1,2,3},{4,5,6,7},{8,9,10,11}};
```

Arr[5][4]

Arr[0][0]=0	Arr[0][1]=1	Arr[0][2]=2	Arr[0][3]=3
Arr[1][0]=4	Arr[1][1]=5	Arr[1][2]=6	Arr[1][3]=7
Arr[2][0]=8	Arr[2][1]=9	Arr[2][2]=10	Arr[2][3]=11
Arr[3][0]	…	…	…
Arr[4][0]	…	…	…

Array

- 二維陣列的基本IO



```
int arr[3][2] = {0};//宣告一個有3列2行的二維陣列，初始值都是0
for( int i = 0 ; i < 3 ; i++)
{
    for( int j = 0 ; j < 2 ; ++j)
    {
        scanf("%d",&arr[i][j]);
    }
}
for( int i = 0 ; i < 3 ; i++)
{
    for( int j = 0 ; j < 2 ; ++j)
    {
        printf("%d ",arr[i][j]);
    }
    printf("\n");//方便辨識
}
```

Array

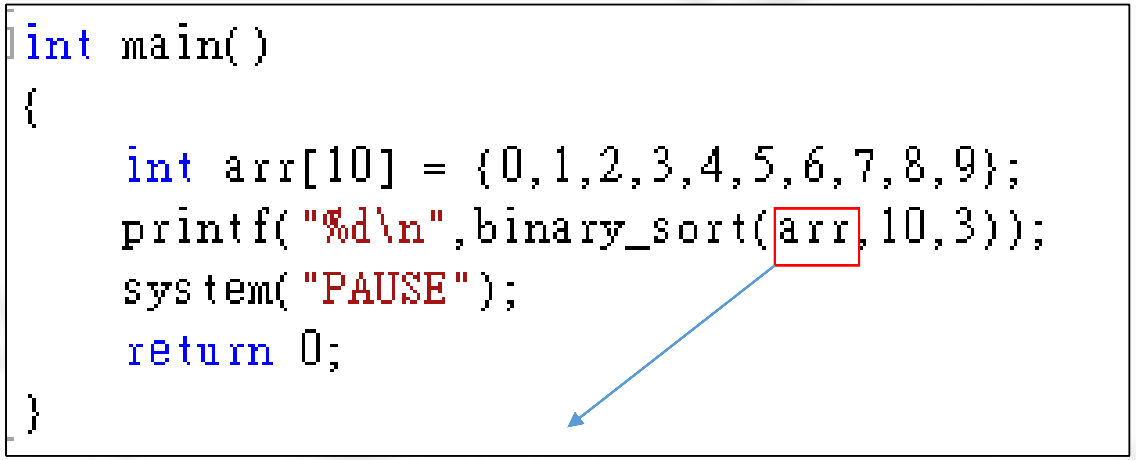
陣列的參數傳遞

- 以~~Pass By Adress~~的方式，資料會一起被更改。
- 陣列的開頭就是陣列的記憶體起始位置。
(arr[0] 和 arr的記憶體位置一樣)

	0x0000	0x0004	0x0008	0x000C
Int arr[4]	Arr[0]	Arr[1]	Arr[2]	Arr[3]

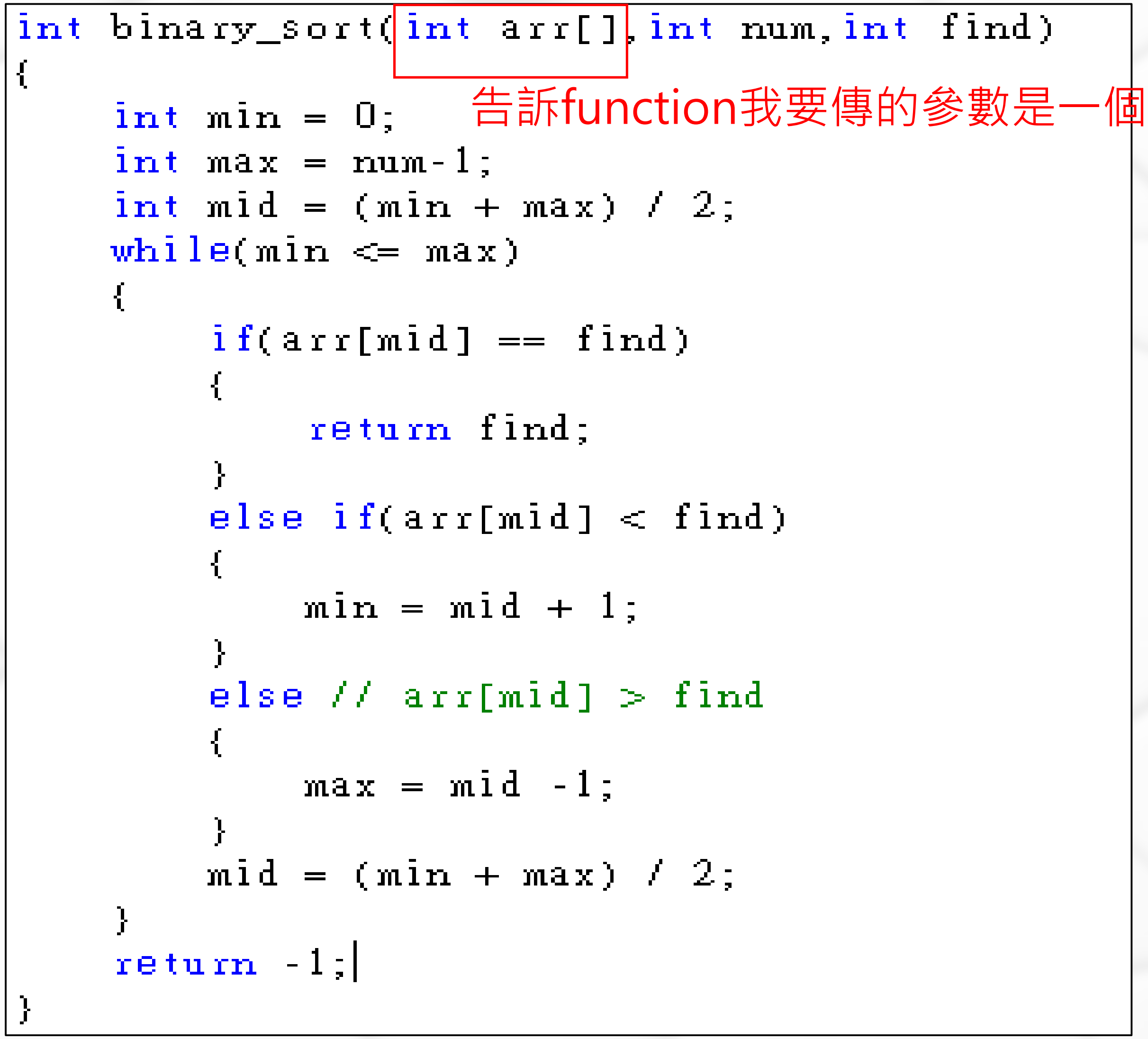
- 一維陣列的參數傳遞需要告知起始位置。
- 二維陣列的參數傳遞需要告知起始位置、col數。

Binary Search



```
int main()
{
    int arr[10] = {0,1,2,3,4,5,6,7,8,9};
    printf("%d\n",binary_sort(arr,10,3));
    system("PAUSE");
    return 0;
}
```

一維陣列的參數傳遞需要告知起始位置
(arr代表arr[0]的記憶體位址)

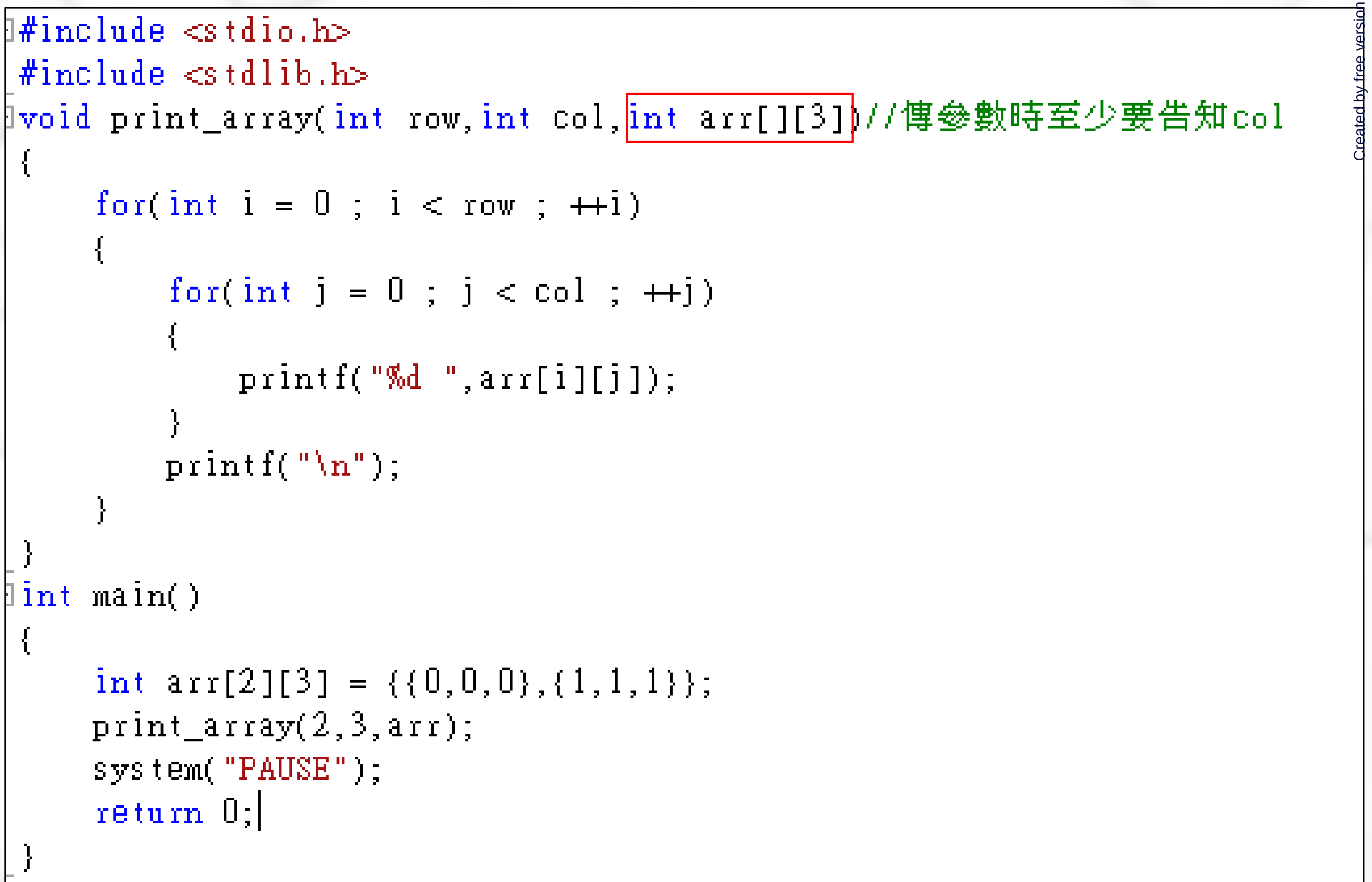


```
int binary_sort(int arr[],int num,int find)
{
    int min = 0;
    int max = num-1;
    int mid = (min + max) / 2;
    while(min <= max)
    {
        if(arr[mid] == find)
        {
            return find;
        }
        else if(arr[mid] < find)
        {
            min = mid + 1;
        }
        else // arr[mid] > find
        {
            max = mid -1;
        }
        mid = (min + max) / 2;
    }
    return -1;
}
```

告訴function我要傳的參數是一個陣列

Array

- 二維陣列的參數傳遞



```c
#include <stdio.h>
#include <stdlib.h>
void print_array(int row,int col,int arr[][3])//傳參數時至少要告知col
{
    for(int i = 0 ; i < row ; ++i)
    {
        for(int j = 0 ; j < col ; ++j)
        {
            printf("%d ",arr[i][j]);
        }
        printf("\n");
    }
}
int main()
{
    int arr[2][3] = {{0,0,0},{1,1,1}};
    print_array(2,3,arr);
    system("PAUSE");
    return 0;
}
```

HW相關小練習

Q：印出質數？

THANK YOU