Chapter 8.

Arrays

I-Fen Chao

Find the nth prime number

怎麼使用function?

```
4   bool is_prime(int x);
5   int get_nth_prime(int num);
6   void print_prime(int num, int step);
7
8   void main(void)
9   {
14
15      system("pause");
16  }
```

```
4   bool is_prime(int x);
5   int get_nth_prime(int num);
6   void print_prime(int num, int step);
7
8   void main(void)
9   {
10      printf("The 100th prime is: %d\n", get_nth_prime(100));
11      printf("The 700th prime is: %d\n", get_nth_prime(700));
12
13      print_prime(100, 10);
14
15      system("pause");
16  }
```

```
18  bool is_prime(int x)
19  {
20      int i;
21      if (x == 1)
22          return false;
23      for (i = 2; i < x; i++) {
24
25                              //not a prime
26      }
27      return true;
28  }
```

```
bool is_prime(int x);
int get_nth_prime(int num);
void print_prime(int num, int ste
```

=

```
bool is_prime(int x)
{
    int i;
    if (x == 1)
        return false;
    for (i = 2; i < x; i++) {
        if (x%i == 0)
            return false;//not a prime
    }
    return true;
}
```

```
int get_nth_prime(int num)
{
    int count = 0; //計算目前prime的數量
    int i = 2; //從i=2開始找prime
    while (                ) {
        if (is_prime(i) == true)
            count++;
        i++;
    }
    return
}
```

```c
bool is_prime(int x);
int get_nth_prime(int num);
void print_prime(int num, int step);
```

```c
int get_nth_prime(int num)
{
    int count = 0; //計算目前prime的數量
    int i = 2; //從i=2開始找prime
    while (count != num) {
        if (is_prime(i) == true)
            count++;
        i++;
    }
    return i - 1;
}
```

```c
void print_prime(int num, int step) //num: 印幾個prime; step:逢step個prime換行
{
    int i=2;
    int count = 0; //累計目前prime的數量
    //for (i = 2; i <= num; i++)
    while(          )
    {
        if (is_prime(i) == true){
            printf("%3d\t", i);
            count++;


        }
        i++;
    }
}
```

```c
bool is_prime(int x);
int get_nth_prime(int num);
void print_prime(int num, int step);
```

```c
void print_prime(int num, int step) //num: 印幾個prime; step:逢step個prime換行
{
    int i=2;
    int count = 0; //累計目前prime的數量
    //for (i = 2; i <= num; i++)
    while(count<num)
    {
        if (is_prime(i) == true){
            printf("%3d\t", i);
            count++;
            if (count%step==0)
                printf("\n");
        }
        i++;
    }
}
```

Array

- Array: a set of the same data type

一維陣列的宣告格式

資料型態 陣列名稱[個數];

```
element_type array_name [number_of_element];
```

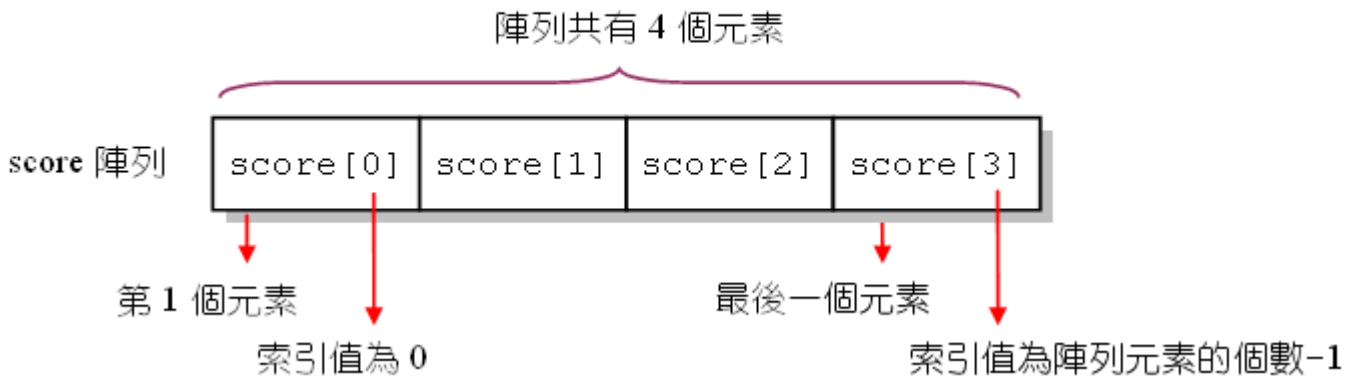
- Declare arrays：

```
int score[4];    /* 宣告整數陣列score，4 integers */
float temp[7];   /* 宣告浮點數陣列temp，7 float numbers */
char name[12];   /* 宣告字元陣列name，12 characters (a string) */
```

Index of an array

- Array index to specify the location within the array
- Array index begin at “0 “

```
int score[4];
```



First index = 0

Last index = size -1

Initialization of an array

一維陣列初值設定的格式

資料型態 陣列名稱[個數n]={初值1,初值2,…,初值n};

data_type *array_name*[*number_of_element*]={**value1, value2,…**};

- Examples：
  - `int score[4]={78,55,92,80};`
  - `int score[]={60,75,48,92};  /* ignore number of elements, this example allocates 4 elements*/`
  - `int data[5]={0};  /*all 0 for 5 elements */`

An Example

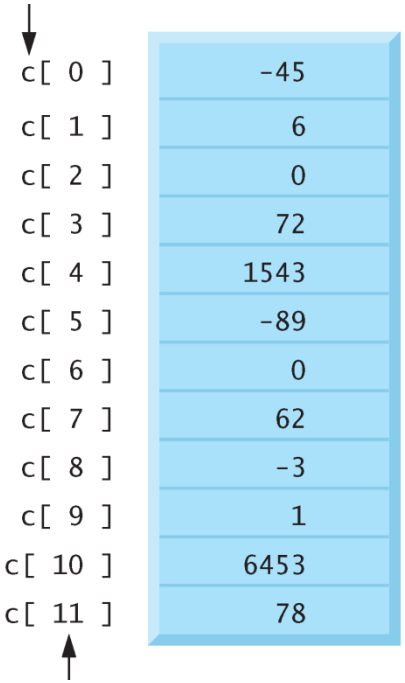
不要加分號”;”

```
/* Program for Lesson 6_1 */
#define N 10
void main(void)
{
   int a[2];
   double b[N];

   a[0]=11;
   a[1]=22;

   b[3]=777.7;
   b[6]=888.8;

   printf("a[0] = %3d, a[1] = %3d\n", a[0],a[1]);
   printf("b[3] = %8.2lf, b[6] = %8.2lf \n", b[3],b[6]);
   printf("b[2] = %lf\n", b[2]);
   printf("a[3] = %d\n", a[3]);
}
```

1-D array

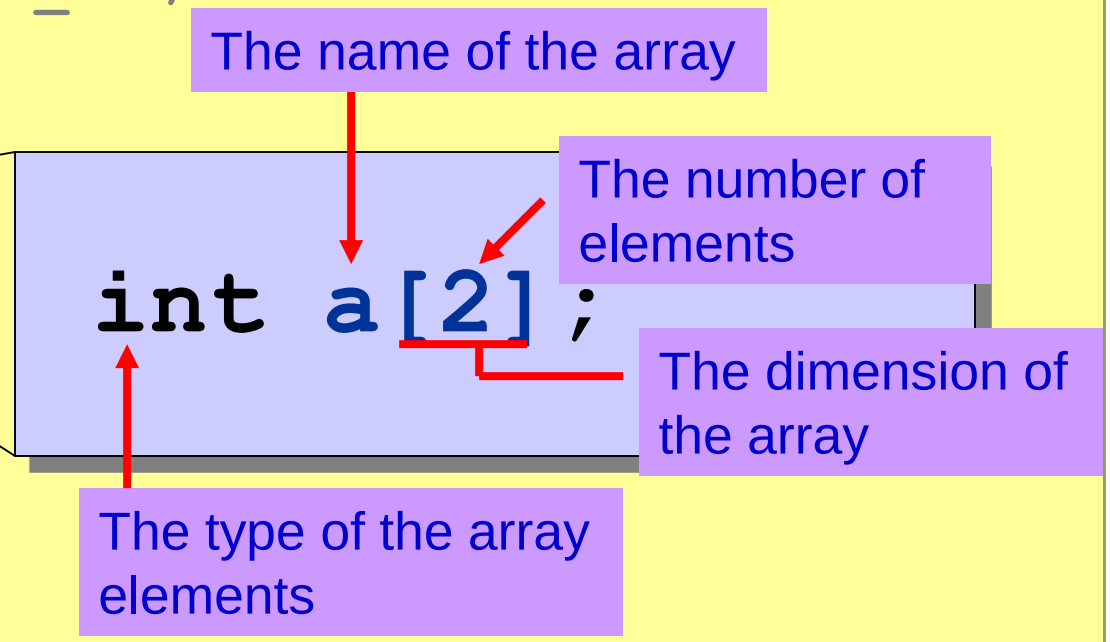
- A one-dimensional array is a collection of <u>the same type of variables</u> stored in *contiguous and increasing* memory locations.
- It is identified by its *name*, *type*, *dimension*, and *number of elements*.
  - e.g. `int c[12];`

Name of array (Note that all elements of this array have the same name, c)



Position number of the element within array c

```
/* Program for Lesson 6_1 */
#define N 10
void main(void)
{
   int a[2];
   double b[N];

   a[0]=11;
   a[1]=22;

   b[3]=777.7;
   b[6]=888.8;

   printf("a[0] = %3d, a[1] = %3d\n", a[0],a[1]);
   printf("b[3] = %8.2lf, b[6] = %8.2lf \n", b[3],b[6]);
   printf("b[2] = %lf\n", b[2]);
   printf("a[3] = %d\n", a[3]);
}
```



The declaration of 1-D array

- Syntax

classified as identifier

```
element_type array_name[number_of_element];
```

any valid C data type, except void and function type

this value must be equal to a positive integer

The dimension of an array

- 1-D array

```
int aa[2];
```

- 2-D array

```
int bb[2][5];
```

- 3-D array

```
int cc[2][2][4];
```

The length of an array

- The length of an array
  - The number of elements
- The length of an array is defined as follow

```
int a[2], c[200], g[100];
```

```
#define  N  10
int     b[N];
double c[N+5], x;
```

```
int c[-25], b[32.5];
```

The amount of memory reserved

- The amount of memory reserved for a one-dimensional array

the number of elements × the amount of memory for each element

For example:

```
float b[10];
→ [4 bytes/elements] x [10 elements] = 40 bytes
```

The subscript of an array

- The first index of an array in C
  - 「0」

```
int a[2];

a[0] = 11;
a[1] = 22;
```

a[2] ← *means the third element of the array* **a[]**

C does not check to see if you try to access an array outside of its range

To access the array elements

- We can treat an array element like we treat a single variable.

array element ⟷ single variable

```
printf("a[0] = %3d, a[1] = %3d\n", a[0],a[1]);
printf("b[3] = %8.2lf, b[6] = %8.2lf \n", b[3],b[6]);
printf("b[2] = %lf\n", b[2]);
printf("a[3] = %d\n", a[3]);
```

The Characteristics of an Array

- Group of consecutive memory locations
- Same name and type
- First element at position 0
  - e.g. n element array named c: c[0], c[1]...c[n – 1]
  - c also represents the start address of the array
- Array elements are like normal variables
  - e.g. c[0] = 3; printf("%d", c[0]);
- Perform operations in subscript
  - c[5 - 2], c[3], c[x]
- Defining multiple arrays of same type
  - int b[100], x[27];
- Arrays have no bounds checking in C programming
  - int a[2]; a[2]=100; ➔ C not detect it, but may cause error in your program

c[0]	-45
c[1]	6
c[2]	0
c[3]	72
c[4]	1543
c[5]	-89
c[6]	0
c[7]	62
c[8]	-3
c[9]	1
c[10]	6453
c[11]	78

Array Initialization

- Methods for initializing array elements in declarations
- Initializing array elements using **`scanf()`**
- Initializing array elements using assignment statements in loops

An example- 1-D array

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
   int i,score[4];

   score[0]=78;
   score[1]=55;
   score[2]=92;
   score[3]=80;

   for(i=0;i<=3;i++)    // for(i=0;i<4;i++)
      printf("score[%d]=%d\n",i,score[i]);

   system("pause");
   return 0;
}
```

scanf- 1-D array

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
   int i,age[3];
   for(i=0;i<3;i++)
   {
      printf("Enter age[%d]:",i);
      scanf("%d",&age[i]);    /*read from keyboard to the array*/
   }
   for(i=0;i<3;i++)
      printf("age[%d]=%d\n",i,age[i]);

   system("pause");
   return 0;
}
```

Initialize the elements of a 1-D array

```
int a[3] = {11,22};
```

type name **[***number_of_elements***]** **=** **{** *value0*, *value1*, ... **}**

```
int b[] = {44,55,66};
```

type name **[]** **=** **{** *value_0*, *value_1*, ..., *value_n* **}**

The number of elements in the array is n+1

Something remains

```
double y[20];
...
y = 100.0;
```

error

In C, we must initialize each element individually. We must modify the array element by element.

```
int b[2] = {44, 55, 66};
```

Although C will not detect out-of-range errors with arrays, it will detect this error

Department of Electrical Engineering, Yuan Ze University

Array Initialization

- Array initialization:
  - int n[5] = { 1, 2, 3, 4, 5 };
  - If not enough initializers, rightmost elements become 0

    e.g. int n[5] = { 0 }; //all elements are 0
- If size omitted, initializers determine the size
  - e.g. int n[ ] = { 1, 2, 3, 4, 5, 6 };

    //6 initializers, therefore 6 element array

Pass "array" as Function parameter

```
/* prog9_12 OUTPUT---
陣列的內容為: 5 3 6 1
--------------------*/
```

傳入的是陣列的**地址**

```
01  /* prog9_12, 傳遞一維陣列到函數裡 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  #define SIZE 4
05  void show(int arr[]);          /* 宣告函數 show()的原型 */
06  int main(void)
07  {
08     int A[SIZE]={5,3,6,1};      /* 設定陣列 A 的初值 */
09     printf("陣列的內容為: ");
10     show(A);                    /* 呼叫函數 show() */
11     system("pause");
12     return 0;
13  }
14  void show(int arr[])           /* 函數 show()的定義 */
15  {
16     int i;
17     for(i=0;i<SIZE;i++)
18        printf("%d ",arr[i]);    /* 印出陣列內容 */
19     printf("\n");
20  }
```

Address of an array

```
/* prog9_15 OUTPUT----
A[0]=20,位址=0022FF48
A[1]= 8,位址=0022FF4C
A[2]=13,位址=0022FF50
陣列A的位址=0022FF48
------------------------*/
```

- Array name is the address of the beginning of the array, and it is the same with the first index of the array.

e.g. A, and A[0] have the same address

```
01  /* prog9_15, 印出陣列的位址 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  #define SIZE 3
05  int main(void)
06  {
07     int i,A[SIZE]={20,8,13};
08     for(i=0;i<SIZE;i++)
09        printf("A[%d]=%2d,位址為%p\n",i,A[i],&A[i]);
10     printf("陣列A的位址=%p\n",A);
11     system("pause");
12     return 0;
13  }
```

A practice :
Add 2 to each element in the array

```
02  #include <stdio.h>
03  #include <stdlib.h>
04  #define SIZE 4
05  void show(int arr[]);
06  void add2(int arr[]);
07
08  int main(void)
09  {
10     int A[SIZE]={5,3,6,1};
```

```
/* prog9_16 OUTPUT------------

呼叫 add()前,陣列的內容為: 5 3 6 1
呼叫 add()後,陣列的內容為: 7 5 8 3
------------------------------*/
```