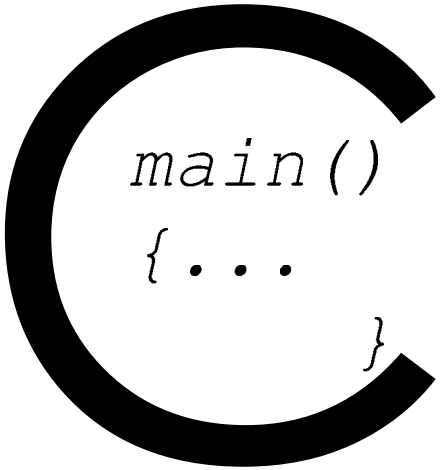
C Programming Practice (32H)

Yi Chen
leo.chen.yi@live.co.uk

- **No1** - Introduction (2H)
- **No2** - Types, Operators And Expressions (2H)
- **No3** - Control Flow (6H)
  - Statements and Blocks
  - Conditional Statements
    - If-Else
    - Switch
  - Loop
    - While
    - For
- **No4** - Functions And Program Structure (6H)
- **No5** - Pointers And Arrays (4H)
- **No6** - Case Studies And Practices (12H)

C Programming Practice
No[5-2]

Chen , Yi
leo.chen.yi@live.co.uk
30-Jul-2010



Arrays

BETA 1.0.0.1

Contents

One-Dimensional Arrays
Input and Output of Array Values
Array Initialisation
Two-Dimensional Arrays
Larger-Dimensional Arrays
Applications
  1: Curve Plotting
  2: Data Scalling
Common Programming Errors
Enrichment Study: Sorting Methods

Contents

à One-Dimensional Arrays
Input and Output of Array Values
Array Initialisation
Two-Dimensional Arrays
Larger-Dimensional Arrays
Applications
  1: Curve Plotting
  2: Data Scalling
Common Programming Errors
Enrichment Study: Sorting Methods

One-Dimensional Arrays

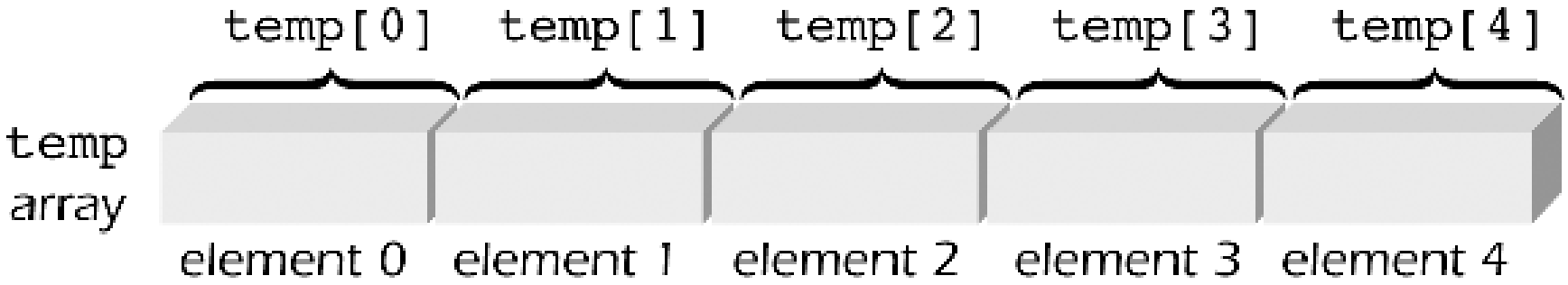
```
char name [10] = {'1','2'}
char name []   = "skyhook"
```

One-dimensional array: a list of related values with the same data type, stored using a single group name (called the **array name**)

Syntax:

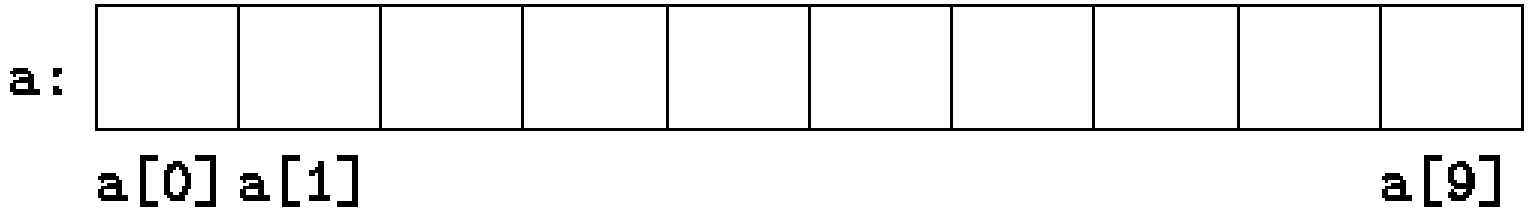
dataType arrayName[number-of-items]

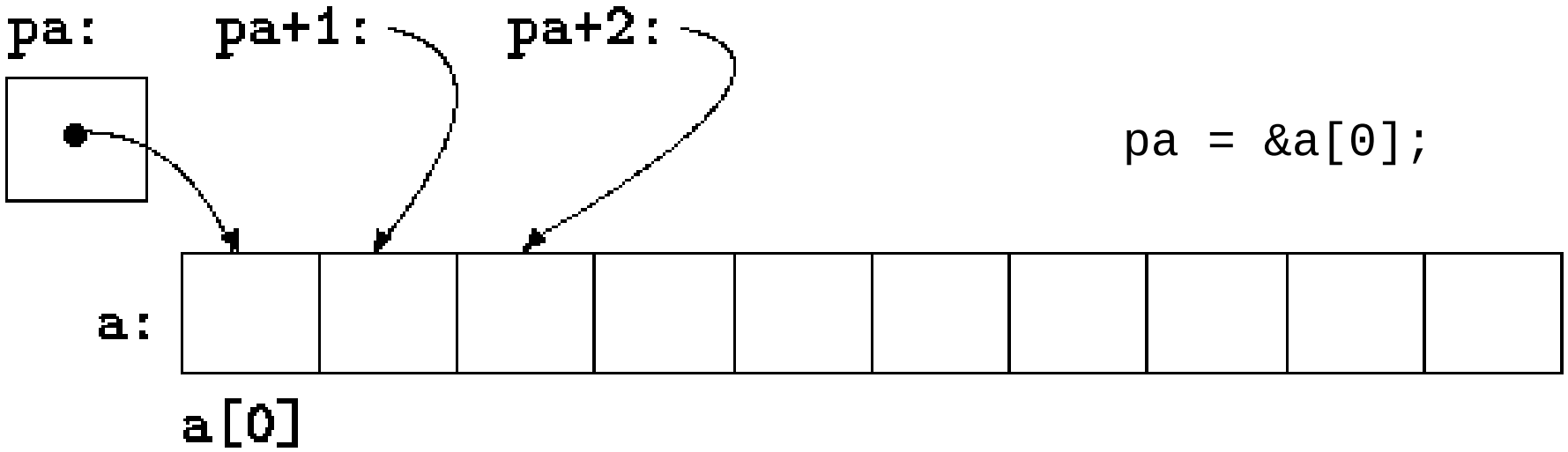
char name[10] = "Hello!"



One-Dimensional Arrays

```
int a[10];
```






```
pa = &a[0];
```

Contents

One-Dimensional Arrays

à Input and Output of Array Values

Array Initialisation

Two-Dimensional Arrays

Larger-Dimensional Arrays

Applications

  1: Curve Plotting

  2: Data Scalling

Common Programming Errors

Enrichment Study: Sorting Methods

Input and Output of Array Values

Input

```
scanf("%f",        &temp[0]);
scanf("%f %f %f", &temp[l], &temp[2], temp[3]);
scanf("%f %f",    &temp[4], &volts[6]);
```

```
for (ind = 0; ind < 4; ind++)
{
  printf("Enter a temperature:");

  scanf("%f",        &temp[i]);

}
```

Input and Output of Array Values

Output

```
printf("%f", volts[6]);

printf("The value of element %d isif", ind, temp[ind]);
```

Contents

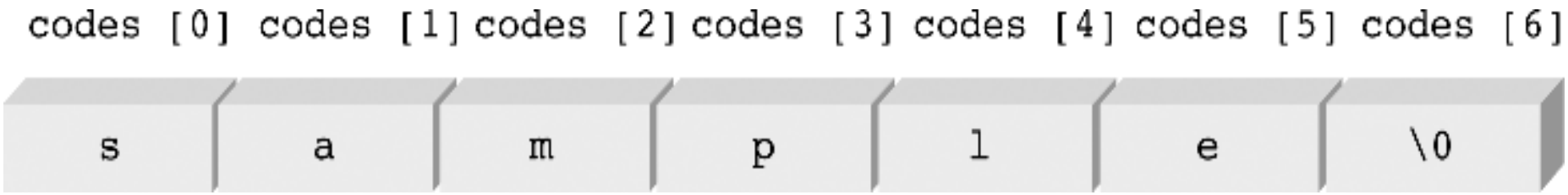
One-Dimensional Arrays
Input and Output of Array Values
à Array Initialisation
Two-Dimensional Arrays
Larger-Dimensional Arrays
Applications
  1: Curve Plotting
  2: Data Scalling
Common Programming Errors
Enrichment Study: Sorting Methods

Array Initialisation

Array elements can be initialized in the array declaration statement

Example:

```
int temp[5] = {98, 87, 92, 79, 85};

char codes[ ] = "sample";
```



Array Initialisation

```
static char *name[] = {
          "Illegal month",
          "January", "February", "March",
          "April", "May", "June",
          "July", "August", "September",
          "October", "November", "December" };
```

Contents

One-Dimensional Arrays
Input and Output of Array Values
Array Initialisation
à Two-Dimensional Arrays
Larger-Dimensional Arrays
Applications
  1: Curve Plotting
  2: Data Scalling
Common Programming Errors
Enrichment Study: Sorting Methods

Two-Dimensional Arrays

- **Two-dimensional array:**

has both rows and columns; also called a **table**

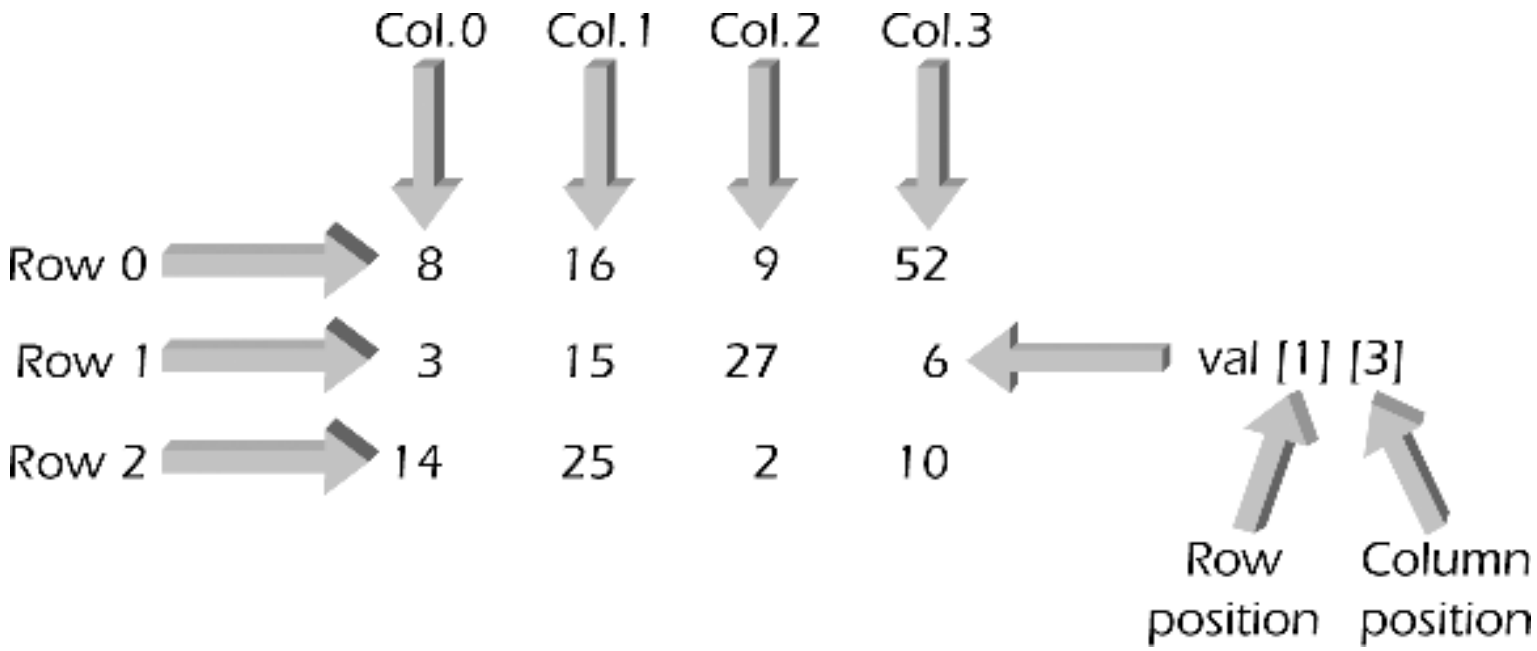
```
static char daytab[2][13] = {
{0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31},
{0, 31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31}
};
```

Two-Dimensional Arrays

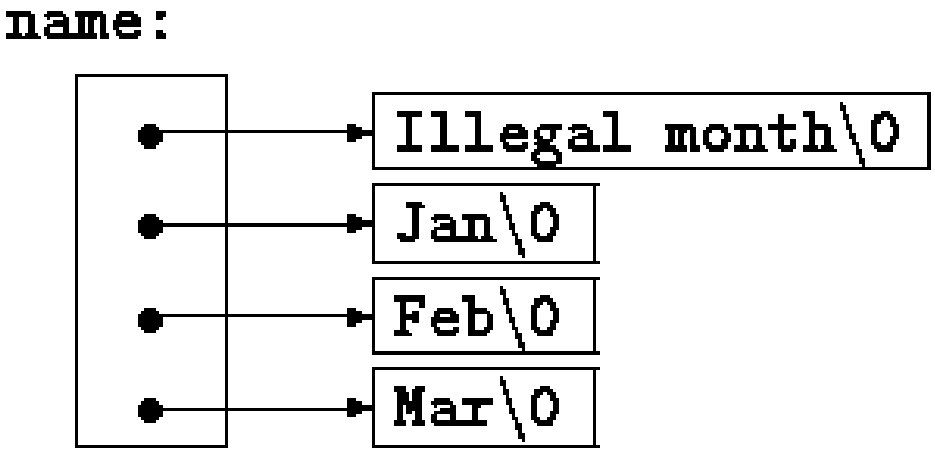
```
daytab[i][j] /* [row][col] */

rather than

daytab[i,j] /* WRONG */
```

Example:

```
int val[1][3];
```



Two-Dimensional Arrays




```
char *name[] = { "Illegal month", "Jan", "Feb", "Mar" };
```

```
char aname[][15] = { "Illegal month", "Jan", "Feb", "Mar" }
```

aname:

Illegal month\0	Jan\0	Feb\0	Mar\0
0	15	30	45

Two-Dimensional Arrays

Initialization starts with this element

```
val[0][0] =  8 --> val[0][1] = 16 --> val[0][2] =  9 --> val[0][3] = 52
                                                                      ↓
        -------<-----------<---------<----------<---------<----------
val[1][0] =  3 --> val[1][1] = 15 --> val[1][2] = 27 --> val[1][3] =  6
                                                                      ↓
        -------<-----------<---------<----------<---------<----------
val[2][0] = 14 --> val[2][1] = 25 --> val[2][2] =  2 --> val[2][3] = 10
```

Contents

One-Dimensional Arrays
Input and Output of Array Values
Array Initialisation
Two-Dimensional Arrays
à Larger-Dimensional Arrays
Applications
  1: Curve Plotting
  2: Data Scalling
Common Programming Errors
Enrichment Study: Sorting Methods

Larger-Dimensional Arrays

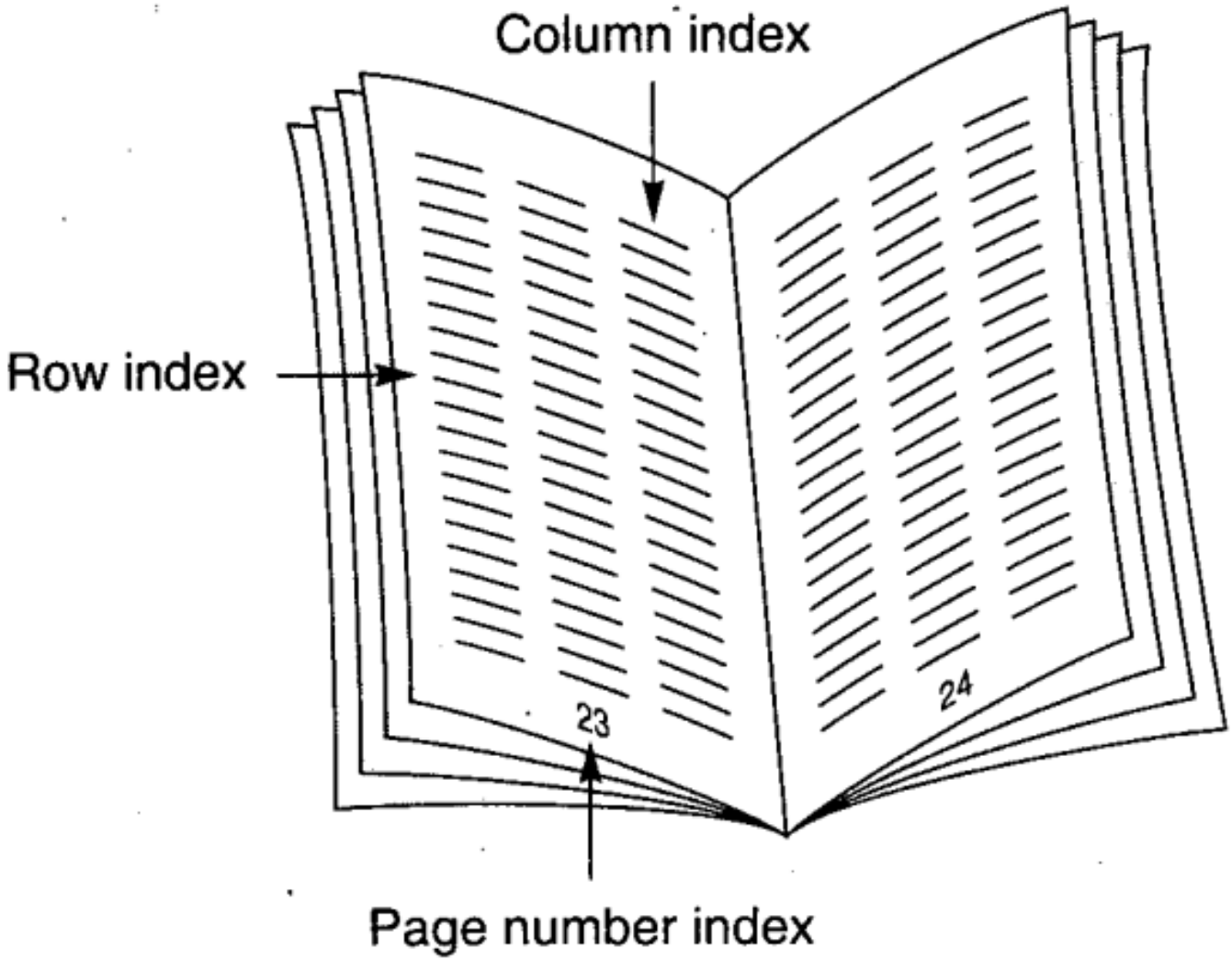
Although arrays with more than two dimensions are not commonly used, C does allow any number of dimensions to be declared.

This is done by listing the maximum size of all dimensions for the array.

For example, the declaration ***int response[4][10][6];*** declares a **three-dimensional** array.

The first element in the array is designated as ***response[0][0][0]*** and the last element as ***response[3][9][5].***

Larger-Dimensional Arrays



Contents

One-Dimensional Arrays
Input and Output of Array Values
Array Initialisation
Two-Dimensional Arrays
Larger-Dimensional Arrays
à Applications
  1: Curve Plotting
  2: Data Scalling
Common Programming Errors
Enrichment Study: Sorting Methods

Application 1: Curve Plotting

```
                                y axis
+------------------------------------------------------------------------------>
```

the actual graph of the data points consists of 15 individual lines, as follows:

```
line 1:  |                                                                    *
line 2:  |                                                  *
line 3:  |                                      *
line 4:  |                          *
line 5:  |                  *
line 6:  |          *
line 7:  |      *
line 8:  |  *
line 9:  |      *
line 10: |          *
line 11: |                  *
line 12: |                          *
line 13: |                                      *
line 14: |                                                  *
line 15: |                                                                    *
```

Application 1: Curve Plotting

Step 1. Store an asterisk in the desired array element.

Step 2. Print the array.

Step 3. Reset the element to a blank.

Step 4. Repeat Steps 1 through 3 until the required number of lines have been displayed.

Application 1: Curve Plotting

```
for ( xdata = 1; xdata <= 15;  ++xdata )
{
/* calculate a value for ydata */
y = pow((x-8),2.0) + 3;

/* set character to an asterisk */
line[y] = '*';

printf("\n%s",line) ;

/* reset character to a blank  */
line[ydata] = [];
                                                    }
```

Application 1: Curve Plotting

```
#include   <stdio.h>
#include   <math.h>
main(  )
{
int  xdata,ydata;
char label[] = "y  axis";
char axis[]  = "+-----------------------------
--------------------------->";
char line[]  = "|
";
printf("\n%s" ,label);
printf("\n%s" ,axis);



...
}
```

Applications

Application 2: Data Scaling

A common problem encountered in plotting data is the need to scale values to fit within the width of the paper or video screen before a plotting routine can be used.

$$\text{Scaled value} = \boxed{\frac{\text{Original value} - \text{Minimum value}}{\text{Maximum value} - \text{Minimum value}}} x$$

where the Maximum and Minimum values are the respective maximum and minimum values for the complete set of data values being plotted.

Contents

One-Dimensional Arrays
Input and Output of Array Values
Array Initialisation
Two-Dimensional Arrays
Larger-Dimensional Arrays
Applications
  1: Curve Plotting
  2: Data Scalling
à Common Programming Errors
Enrichment Study: Sorting Methods

Common Programming Errors

1. Forgetting to declare the array. This error results in a compiler error message equivalent to "invalid indirection" each time a subscripted variable is encountered within a program.

2. Using a subscript that references a nonexistent array element. For example, declaring the array to be of size 20 and using a subscript value of 25. This error is not detected by most C compilers. It can, however, result in a run-time error that results in a program "crash" or a value that has no relation to the intended array element. In either case this is usually an extremely troublesome error to locate. The only solution to this problem is to make sure, either by specific programming statements or by careful coding, that each subscript references a valid array element.

Common Programming Errors

3. Not using a large enough conditional value in a `for` loop counter to cycle through all the array elements. This error usually occurs when an array is initially specified to be of size `n` and there is a `for` loop within the program of the form `for (i = 0; i < n; ++i)`. The array size is then expanded but the programmer forgets to change the interior for loop parameters.

4. Forgetting to initialize the array. Although many compilers automatically set all elements of integer and real valued arrays to zero and all elements of character arrays to blanks, it is up to the programmer to ensure that each array is correctly initialized before processing of array elements begins.

Contents

One-Dimensional Arrays
Input and Output of Array Values
Array Initialisation
Two-Dimensional Arrays
Larger-Dimensional Arrays
Applications
  1: Curve Plotting
  2: Data Scalling
Common Programming Errors
à Enrichment Study: Sorting Methods

Sorting Methods

selection sorting

insertion sorting

merge sorting

quicksort

http://en.wikipedia.org/wiki/Sorting_algorithm

C Programming Practice
No[5]

Chen , Yi
leo.chen.yi@live.co.uk
30-Jul-2010

```
main()
{...
}
```

Arrays

End

BETA 1.0.0.1

[太常引.Autumn-Winter]

落花弦上曲散迟，杏黄菊庭时。
何年经往事，墨迹几行词几支？

邂逅长安，风推云幡，月影满霓裳。
琵琶语夜半单，江枫渔火泛微澜。