C Arrays

Arrays

- Arrays
- Defining and Initializing Arrays
- Array Example
- Subscript Out-of-Range Example
- Passing Arrays to Functions
  - Call by Reference
- Multiple-Subscripted Arrays
  - Double-Subscripted Array Example

Arrays

Arrays :: Structures of related data items

- Static entity, namely, the size of an array must be a **constant** throughout the program.
- A group of consecutive memory locations with the same name and type.

Arrays

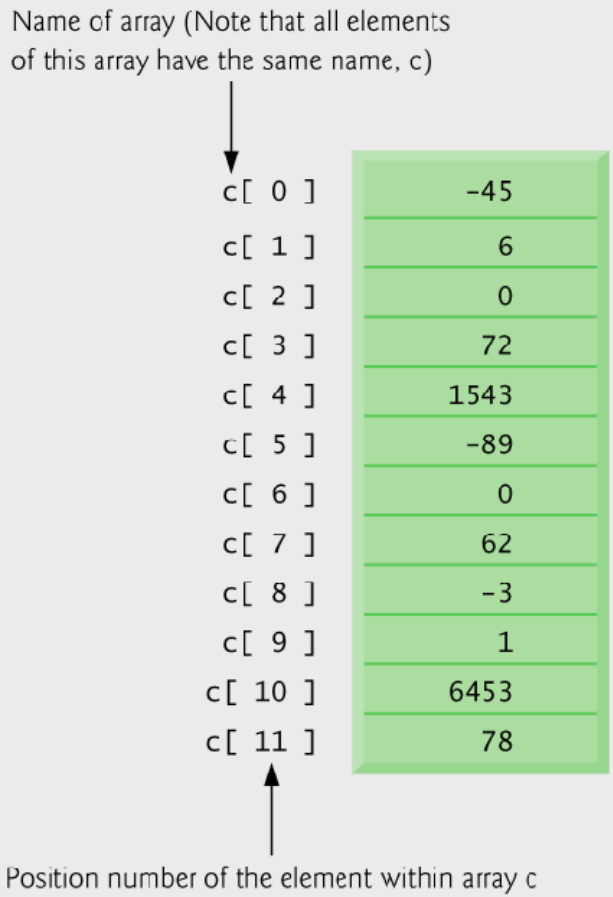
- To refer to an array element, specify
  - the array name
  - the position number {in C this is an offset}
- Format:

  arrayname[*position number*]

  - First element at position `0`
  - n element array named c:

    `c[ 0 ], c[ 1 ]...c[ n - 1 ]`

Fig. 6.1 12-element array



Arrays

- Array elements are like normal variables.

```
c[ 0 ] =  3;
printf( "%d", c[ 0 ] );
```

- One can perform operations within the subscript.

Examples:

```
c[ x- 2 ] = 6;
c[i+j- 4] = c[x-2];
c[intfcn(i)] = 0;
```

6.3 Defining Arrays

- When defining arrays, specify
  - Name
  - Type of array
  - Number of elements

```
arrayType arrayName[ numberOfElements ];
```

  - Examples:

```
int c[ 10 ];
float myArray[ 3284 ];
```

- Defining multiple arrays of same type
  - Format similar to regular variables
  - Example:

```
int b[ 100 ], x[ 27 ];
```

Initializing Array

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

```
int n[ 5 ] = { 0 }
```

    - All elements 0
  - If too many initializers, a syntax error occurs
  - C arrays have no bounds checking!!
- If size omitted, initializers determine it

```
int n[ ] = { 1, 2, 3, 4, 5 };
```

  - 5 initializers, therefore 5 element array

Fig. 6.5 Array Example

```
1  /* Fig. 6.5: fig06_05.c
2     Initialize the elements of array s to the even integers from 2 to 20 */
3  #include <stdio.h>
4  #define SIZE 10 /* maximum size of array */
5
6  /* function main begins program execution */
7  int main( void )
8  {
9     /* symbolic constant SIZE can be used to specify array size
10    int s[ SIZE ]; /* array s has SIZE elements */
11    int j; /* counter */
12
13    for ( j = 0; j < SIZE; j++ ) { /* set the values */
14       s[ j ] = 2 + 2 * j;
15    } /* end for */
16
17    printf( "%s%13s\n", "Element", "Value" );
18
19    /* output contents of array s in tabular format */
20    for ( j = 0; j < SIZE; j++ ) {
21       printf( "%7d%13d\n", j, s[ j ] );
22    } /* end for */
23
24    return 0; /* indicates successful termination */
25
26 } /* end main */
```

#define directive tells compiler to replace all instances of the word **SIZE** with **10**

SIZE is replaced with **10** by the compiler, so array **s** has 10 elements

Arrays

Dangerous in C because:

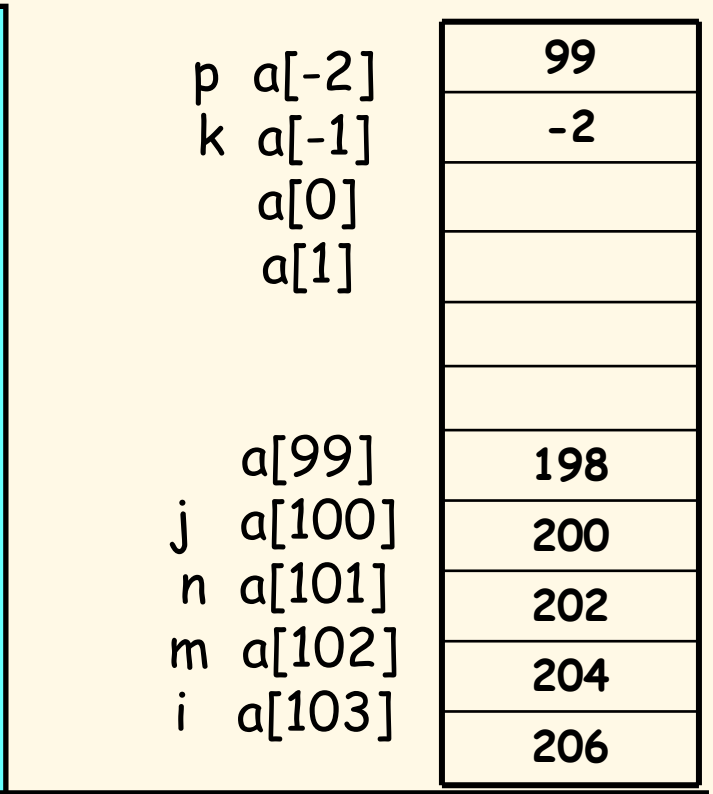
- There is no default initialization.
- There is no bounds checking for subscripts out-of-range.

'Scary' Out-of-Range Example

```
/* Bizarre Example of subscripting out of range */
. int main ()
{
  int i, m , n, j;
  int a[100], k, p;

  j = 77; k = 88; p = 99;

  for (i= -1; i<=103; i++)
  {
      a[i] = 2*i;
      if (i > 98) printf("i =%d, a[i] = %d\n", i, a[i]);
  }
  printf("j = %d, n = %d, m = %d, i = %d, k = %d, p
   = %d\n",
          j, n, m, i, k, p);
  printf("%d %d %d %d %u\n",
       a[-1], a[99], a[100], a[102]
  return 0;
}
```




```
i =99, a[i] = 198
i =100, a[i] = 200
i =101, a[i] = 202
i =102, a[i] = 204
i =206, a[i] = 15
j = 200, n = 202, m = 204, i = 207, k = -2, p = 99
-2 198 200 204 3220526203
```

6.5 Passing Arrays to Functions

- To pass an array argument to a function, specify the name of the array without any brackets.
- The array size is usually passed to the function.

```
int myArray[ 24 ];
myFunction( myArray, 24 );
```

- Arrays are passed **by-reference**.
  - The name of the array is associated with the address of the first array element.
  - The function knows where the array is stored and it can modify the original memory locations.

6.5 Passing Arrays to Functions

- Individual array elements
  - Are passed by value.
  - Pass the subscripted name (i.e., `myArray[ 3 ]`) to function.
- Function prototype

```
void modifyArray( int b[], int arraySize );
```

  - Parameter names are optional in prototype.
    - `int b[]` could be written `int []`
    - `int arraySize` could be simply `int`

Passing Arrays to Functions

```
/* Arrays are passed using Call by Reference */
#include <math.h>
#define SIZE 6
void flip (float fray [], int fsize)
{
  float temp;
  int i,j;

  i = fsize - 1;
  for (j = 0; j < fsize/2 ; j++)
  {
    temp = fray[j];
    fray[j] = fray[i];
    fray[i] = temp;
    i--;
  }
  return;
}
```

Passing Arrays to Functions

```
int main ()
{
  float var[SIZE];
  int i,j;
  for (i=0; i < SIZE; i++)
  {
    var[i] = 1.0/pow (2.0,i);
    printf(" %5.3f", var[i]);
  }
  printf("\n");

  for (j=0; j < 2; j++)
  {
    flip (var, SIZE);
    for (i=0; i < SIZE; i++)
      printf(" %5.3f", var[i]);
    printf("\n");
  }
}
```

```
$./passray
 1.000 0.500 0.250 0.125 0.062 0.031
 0.031 0.062 0.125 0.250 0.500 1.000
 1.000 0.500 0.250 0.125 0.062 0.031
```

6.9 Multiple-Subscripted Arrays

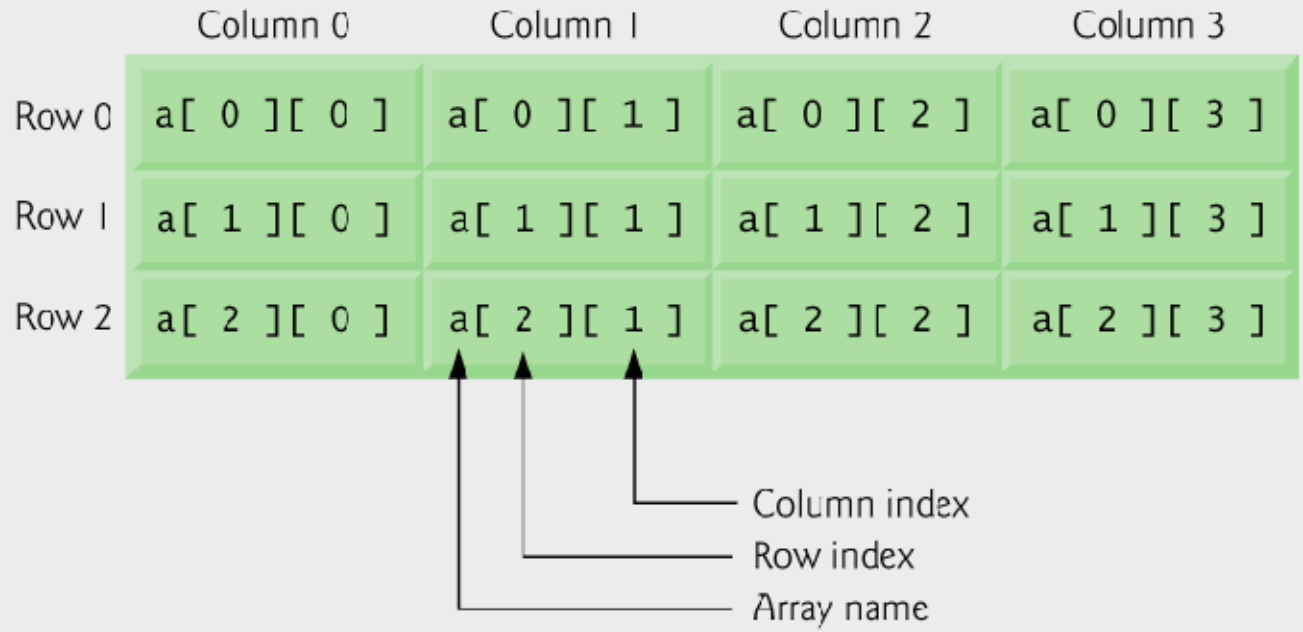
- Multiple subscripted arrays
  - Tables with rows and columns (m by n array)
  - Like matrices: specify row, then column
- Initialization
  - `int b[ 2 ][ 2 ] = { { 1, 2 }, { 3, 4 } };`
  - Initializers grouped by row in braces
  - If not enough, unspecified elements set to zero

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

- Referencing elements
  - Specify row, then column

```
printf( "%d", b[ 0 ][ 1 ] );
```

Fig. 6.20 Double-Subscripted array



- Three rows and four columns.

Double-Subscripted Array Example

```
1  /* Fig. 6.22: fig06_22.c
2     Double-subscripted array example */
3  #include <stdio.h>
4  #define STUDENTS 3
5  #define EXAMS 4
6
7  /* function prototypes */
8  int minimum( const int grades[][ EXAMS ], int pupils, int tests );
9  int maximum( const int grades[][ EXAMS ], int pupils, int tests );
10 double average( const int setOfGrades[], int tests );
11 void printArray( const int grades[][ EXAMS ], int pupils, int tests );
12
13 /* function main begins program execution */
14 int main( void )
15 {
16    int student; /* student counter */
17
18    /* initialize student grades for three students (rows) */
19    const int studentGrades[ STUDENTS ][ EXAMS ] =
20       { { 77, 68, 86, 73 },
21         { 96, 87, 89, 78 },
22         { 70, 90, 86, 81 } };
23
24    /* output array studentGrades */
25    printf( "The array is:\n" );
26    printArray( studentGrades, STUDENTS, EXAMS );
27
```

Each row in the array corresponds to a single student's set of grades

Double-Subscripted Array Example

```
28    /* determine smallest and largest grade values */
29    printf( "\n\nLowest grade: %d\nHighest grade: %d\n",
30       minimum( studentGrades, STUDENTS, EXAMS ),
31       maximum( studentGrades, STUDENTS, EXAMS ) );
32
33    /* calculate average grade for each student */
34    for ( student = 0; student < STUDENTS; student++ ) {
35       printf( "The average grade for student %d is %.2f\n",
36          student, average( studentGrades[ student ], EXAMS ) );
37    } /* end for */
38
39    return 0; /* indicates successful termination */
40
41 } /* end main */
42
```

average function is passed a row of the array

Double-Subscripted Array Example

```
43 /* Find the minimum grade */
44 int minimum( const int grades[][ EXAMS ], int pupils, int tests )
45 {
46    int i; /* student counter */
47    int j; /* exam counter */
48    int lowGrade = 100; /* initialize to highest possible grade */
49
50    /* loop through rows of grades */
51    for ( i = 0; i < pupils; i++ ) {
52
53       /* loop through columns of grades */
54       for ( j = 0; j < tests; j++ ) {
55
56          if ( grades[ i ][ j ] < lowGrade ) {
57             lowGrade = grades[ i ][ j ];
58          } /* end if */
59
60       } /* end inner for */
61
62    } /* end outer for */
63
64    return lowGrade; /* return minimum grade */
65
66 } /* end function minimum */
67
```

Double-Subscripted Array Example

```
68 /* Find the maximum grade */
69 int maximum( const int grades[][ EXAMS ], int pupils, int tests )
70 {
71    int i; /* student counter */
72    int j; /* exam counter */
73    int highGrade = 0; /* initialize to lowest possible grade */
74
75    /* loop through rows of grades */
76    for ( i = 0; i < pupils; i++ ) {
77
78       /* loop through columns of grades */
79       for ( j = 0; j < tests; j++ ) {
80
81          if ( grades[ i ][ j ] > highGrade ) {
82             highGrade = grades[ i ][ j ];
83          } /* end if */
84
85       } /* end inner for */
86
87    } /* end outer for */
88
89    return highGrade; /* return maximum grade */
90
91 } /* end function maximum */
92
```

Double-Subscripted Array Example

```
93 /* Determine the average grade for a particular student */
94 double average( const int setOfGrades[], int tests )
95 {
96    int i; /* exam counter */
97    int total = 0; /* sum of test grades */
98
99    /* total all grades for one student */
100   for ( i = 0; i < tests; i++ ) {
101      total += setOfGrades[ i ];
102   } /* end for */
103
104   return ( double ) total / tests; /* average */
105
106 } /* end function average */
107
108 /* Print the array */
109 void printArray( const int grades[][ EXAMS ], int pupils, int tests )
110 {
111   int i; /* student counter */
112   int j; /* exam counter */
113
114   /* output column heads */
115   printf( "                [0]  [1]  [2]  [3]" );
116
```

Double-Subscripted Array Example

```
117   /* output grades in tabular format */
118   for ( i = 0; i < pupils; i++ ) {
119
120      /* output label for row */
121      printf( "\nstudentGrades[%d] ", i );
122
123      /* output grades for one student */
124      for ( j = 0; j < tests; j++ ) {
125         printf( "%-5d", grades[ i ][ j ] );
126      } /* end inner for */
127
128   } /* end outer for */
129
130} /* end function printArray */
```

```
The array is:
                 [0]  [1]  [2]  [3]
studentGrades[0] 77   68   86   73
studentGrades[1] 96   87   89   78
studentGrades[2] 70   90   86   81

Lowest grade: 68
Highest grade: 96
The average grade for student 0 is 76.00
The average grade for student 1 is 87.50
The average grade for student 2 is 81.75
```

An enum and switch Example

```
/* A program that uses enumerated types, switch and a
   sentinel to terminate input */

#define SENTINEL 10
int main ()
{
  int day;

  /* enum starts assigning positional integers
     beginning with 0 */

  enum days {SUN, MON, TUES, WED, THUR, FRI, SAT};

  scanf("%d", &day);
  while( day != SENTINEL)
  {

     switch (day)
     {
        case MON:
        case WED:
        case FRI:
          printf("%d - Go to class\n", day);
          break;
```

An enum and switch Example

```
    case TUES:
       printf("%d - Sleep in until 10\n", day);
       break;

    case THUR:
       printf("%d - Do laundry\n", day);
       break;

    case SAT:
       printf("%d - Go to gym. ", day);
       printf("Go out to a movie\n");
       break;

    case SUN:
       printf("%d - Study lots!\n", day);
       break;

    default:
       printf("%d - This invalid input. Try again.", day);
       break;
   }
   scanf("%d", &day);
 }
 printf("Sentinal encountered.\n");
 return 0;
}
```