C-7 ARRAYS

One Dimensional Arrays

An Array named seconds:

Contents of the Cell	Name of Each Cell
10	seconds [0]
13	seconds [1]
9	seconds [2]
45	seconds [3]
14	seconds [4]

The array **seconds** contains five values.

Declaring an array:

```
int seconds[5];

float time[100];
```

Initializing the array at start:

```
int seconds[5] = {10, 13, 9, 45, 14};

float time[100] = {0.0};

int a[ ] = {2, 4, 6};
```

Example – Fill an array with values:

```
int j, c = 2, a[100];

for (j = 0; j < 100; j++)
{
      a[ j ] = c;
      c = c + 2;
}
```

Example using a variable for the length of the array

```
#define A_SIZE 100
…
int j, c = 2, a[A_SIZE];
…
for (j = 0; j < A_SIZE; j++)
{
     a[ j ] = c;
     c = c + 2;
}
```

Reading Arrays

```
#define MSIZE 100
…
int c = 0, miles[MSIZE], sum = 0;
FILE *datfile;

datfile = fopen(“travel.dat”, “r”);
if(datfile == NULL)
{     printf(“Error opening file”);
      exit EXIT_FAILURE;
}

while( (fscanf(datfile, “%d”, &miles[c])) == 1)
{
      sum += miles[c];
      c++;
}
```

Printing Arrays – example 1:

```
for (k = 0; k < MSIZE; k++)
{
	printf(“%i \n”, miles[k]);
}
```

Printing Arrays – example 2:

```
for (k = 0; k < c; k++)
{
    printf(“%d \n”, miles[k]);
}
```

Printing Arrays – example 3:

```
for (k = 0; k < MSIZE; k++)
{
     if (k % 4 == 0)
          printf(“\n %f”, miles[k]);
     else
          printf(“%f   “, miles[k]);
}
printf(“\n”);


// Prints 4 numbers per line.
```

Precedence of [ ] :

[ ] appears high on chart with ( )

Both are at the same level

```
/*------------------------------------------------------*/
/* try initialization of array with 2 values      */

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

int main(void)
{
      int i;                    /* loop counter */
      int a[10] = {1, 2};

      for (i = 0; i < 10; i++)
            printf("\nPosition %i has %i", i, a[i]);
       printf("\n\n");

       return EXIT_SUCCESS;
}
```

The run looks like this:

```
Position 0 has 1
Position 1 has 2
Position 2 has 0
Position 3 has 0
Position 4 has 0
Position 5 has 0
Position 6 has 0
Position 7 has 0
Position 8 has 0
Position 9 has 0
```

If part of an array is initialized,
the remaining array is initialized to zero.

Functions & Arrays

Function Arguments and Arrays

Arrays are not the same as simple variables.

Simple variables in function call are call-by-value

Value of variables passed to function

(xerox copy concept)

Arrays in function call are call-by-address

The starting address of the array is passed to function

Not just the values!

The function can **change** the values in the array.

(original copy concept)

Just like what we did with the (& *) pair

Function Prototype:

double max (double x[ ], int n);

where x is an array and n is its size

In main, declare an array and use it in this function call:

```
double y[N];    /* declare array of size N */
int npts;       /* actual length of array */


printf("The maximum value is: %f \n", max(y, npts));
```

Array Examples

```
int x[10] = {-5, 4, 3};
```

If initializing sequence is shorter than the array, then the rest of the values are initialized to zero.

Values of x: -5, 4, 3, 0, 0, 0, 0, 0, 0, 0

Show the contents of this array:

```
double z[4];
…
z[1] = -5.5;

z[2] = z[3] = fabs(z[1]);
```

Show the contents of this array:

```
double z[4];
…
z[1] = -5.5;

z[2] = z[3] = fabs(z[1]);
```

Contents would be:

?   -5.5   5.5   5.5

Show the contents of this array:

```
int k;
double time[9];
…
for (k = 0; k < 9; k++)
{
    time[k] = (k – 4) * 0.1;
}
```

Show the contents of this array:

```
int k;
double time[9];
…
for (k = 0; k < 9; k++)
{
    time[k] = (k – 4) * 0.1;
}
```

Contents would be:

```
-0.4   -0.3   -0.2   -0.1   0.0   0.1   0.2   0.3   0.4
```

Show what is printed for the following:

```
int k, s[ ] = {3, 8, 15, 21, 30, 41};

for (k = 0; k < 6; k += 2)
{
    printf(“%i  %i  \n”, s[k], s[k + 1]);
}
```

Show what is printed for the following:

```
int k, s[ ] = {3, 8, 15, 21, 30, 41};
               0  1   2   3   4   5  ←position in array
for (k = 0; k < 6; k += 2)
{
    printf("%i  %i  \n", s[k], s[k + 1]);
}
```

It would print:

```
 3    8
15   21
30   41
```

Show what is printed for the following:

```
int k, s[ ] = {3, 8, 15, 21, 30, 41};

for (k = 0; k < 6; k++)
{
    if (s[k] % 2 == 0)
    {
        printf(“%i”, s[k]);
    }
}
printf(“\n”);
```

Show what is printed for the following:

```
int k, s[ ] = {3, 8, 15, 21, 30, 41};

for (k = 0; k < 6; k++)
{
    if (s[k] % 2 == 0)
    {
        printf(“%4i”, s[k]);
    }
}
printf(“\n”);
```

```
8  30
```

Two Dimensional Arrays

Two-dimensional Arrays

Arrays with both rows and columns:

Row 0	-1	3	2	6
Row 1	5	3	1	-1
Row 2	10	4	-2	9
	Column 0	***Column 1***	***Column 2***	***Column 3***

Initialization

Row first, then column

```
int c[3][4];    /* the array on the last slide */

int c[3][4] = {{-1, 3, 2, 6}, {5, 3, 1, -1}, {10, 4, -2, 9}};

                /* initialization of the same array */
```

We use nested loops with two-dim. arrays.
What will be in this array when it is finished?

```
int r, c, x[3][4];

for (r = 0; r < 3; r++)
{
    for (c = 0; c < 4; c++)
    {
        x[r][c] = r;
    }
}
```

We use nested loops with two-dim. arrays.
What will be in this array when it is finished?

```
int r, c, x[3][4];

for (r = 0; r < 3; r++)
{
     for (c = 0; c < 4; c++)
          x[r][c] = r;
}
```

```
0 0 0 0
1 1 1 1
2 2 2 2
```

Code to fill an Identity Matrix:

```
int r, c, m[4][4];

for (r = 0; r < 4; r++)
{
    for (c = 0; c < 4; c++)
    {
        if (r == c)
            m[r][c] = 1;
        else
            m[r][c] = 0;
     }
}
```

Code to fill an Identity Matrix:

```
int r, c, m[4][4];

for (r = 0; r < 4; r++)
{
     for (c = 0; c < 4; c++)
     {
          if (r == c)
              m[r][c] = 1;
          else
              m[r][c] = 0;
     }
}
```

$$\begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$

Code to add 2 matrices

```
/*-----------------------------------------------*/
#define N 4      /*parts of main */
void matrix_add(int d[N][N], int e[N][N], int f[N][N]);

int main(void)
{   int r, c;
    int d[N][N], e[N][N], f[N][N];
…
    matrix_add(d, e, f);
    /* loop to print matrix c */
    for (r = 0; r < N; r++)
    {
        for (c = 0; c < N; c++)
             printf(“%i  “, f[r][c]);
     }
}
/*----end of main------------------------------------*/
```

```
/*-------------------------------------------------*/
/* function to add 2 matrices                  */
void matrix_add(int d[N][N], int e[N][N], int f[N][N])
{   int r, c;
    for (r = 0; r < N; r++)
    {
        for (c = 0; c < N; c++)
        {
            f[r][c] = d[r][c] + e[r][c];
        }
    }
    return;         /* void return */
}
/*----end of matrix_add---------------------------*/
```

Show the contents of these arrays

Show the contents of the array:

```
int d[3][1] = {{1}, {4}, {6}};
```

Show the contents of the array:

```
int d[3][1] = {{1}, {4}, {6}};
```

1
4
6

Show the contents of the array:

```
int g[6][2] = {{5,2}, {-2,3}};
```

Show the contents of the array:

```
int g[6][2] = {{5,2}, {-2,3}};
```

```
 5   2
-2   3
 0   0
 0   0
 0   0
 0   0
```

Show the contents of the array:

```
float h[4][4] = {{0.0}};
```

Show the contents of the array:

```
float h[4][4] = {{0.0}};
```

```
0.0  0.0  0.0  0.0
0.0  0.0  0.0  0.0
0.0  0.0  0.0  0.0
0.0  0.0  0.0  0.0
```

Show the contents of the array:

```
int k, p[3][3] = {{0,0,0}};
…
for (k = 0; k < 3; k++)
{
    p[k][k] = 1;
}
```

Show the contents of the array:

```
int k, p[3][3] = {{0,0,0}};
…
for (k = 0; k < 3; k++)
{
    p[k][k] = 1;
}
```

```
1  0  0
0  1  0
0  0  1
```

Show the contents of the array:

```
int r, c, g[5][5];
…
for (r = 0; r < 5; r++)
{
    for (c = 0; c < 5; c++)
    {
        g[r][c] = r + c;
    }
}
```

Show the contents of the array:

```
int r, c, g[5][5];
…
for (r = 0; r < 5; r++)
{
    for (c = 0; c < 5; c++)
    {
        g[r][c] = r + c;
    }
}
```

```
0 1 2 3 4
1 2 3 4 5
2 3 4 5 6
3 4 5 6 7
4 5 6 7 8
```

Show the contents of the array:

```
int r, c, g[5][5];
…
for (r = 0; r < 5; r++)
{
    for (c = 0; c < 5; c++)
    {
        g[r][c] = pow(-1, c);
    }
}
```

Show the contents of the array:

```
int r, c, g[5][5];
…
for (r = 0; r < 5; r++)
{
    for (c = 0; c < 5; c++)
    {
        g[r][c] = pow(-1, c);
    }
}
```

```
1  -1  1  -1  1
1  -1  1  -1  1
1  -1  1  -1  1
1  -1  1  -1  1
1  -1  1  -1  1
```

C-7 ARRAYS

THE END