

## CSE1301 Computer Programming Lecture 19 Arrays (Part 2)

1

## Topics

- Two-dimensional arrays
- Passing two-dimensional arrays to functions

2

## Two-dimensional Arrays

- Each element of an array is like a single item of a particular type
- But an array itself is an item of a particular type  
➔ So, an array element could be another array
- An “array-of-arrays” is called “multi-dimensional” because you need to specify several ordinates to locate an actual element

3

## Example: YearlyRainfall

	month												
	0	1	2	3	4	5	6	7	8	9	10	11	
year	0	30	40	75	95	130	220	210	185	135	80	40	45
1	25	25	80	75	115	270	200	165	85	5	10	0	
2	35	45	90	80	100	205	135	140	170	75	60	95	
3	30	40	70	70	90	180	180	210	145	35	85	80	
4	30	35	30	90	150	230	305	295	60	95	80	30	

*Average Yearly Rainfall (in mm)*

Problem: using the *Yearly Rainfall* table

- input month and year
- output mean rainfall for that month and year

4

## Example (cont): YearlyRainfall-1

```

#define NYEARS 5
#define NMONTHS 12

int lookup(int year, int month)
{
    int table[NYEARS][NMONTHS] =
    {
        {30,40,75,95,130,220,210,185,135,80,40,45},
        {25,25,80,75,115,270,200,165, 85, 5,10, 0},
        {35,45,90,80,100,205,135,140,170,75,60,95},
        {30,40,70,70, 90,180,180,210,145,35,85,80},
        {30,35,30,90,150,230,305,295, 60,95,80,30}
    };

    if ((0 <= year) && (year < NYEARS) &&
        (0 <= month) && (month < NMONTHS))
    {
        return table[year][month];
    }
    else
    {
        return -1;
    }
}
    
```

yearly1.c  
5

## Example (cont): YearlyRainfall-2

```

int main()
{
    int year;
    int month;
    int rainfall;

    printf("Enter year and month: ");
    scanf("%d %d", &year, &month);

    rainfall = lookup(year - 1, month - 1);

    if (rainfall < 0)
    {
        printf("Year must be between 1 and %d.\n", NYEARS);
        printf("and month must be between 1 and %d.\n", NMONTHS);
    }
    else
    {
        printf("Rainfall for year %d, month %d is %d mm.\n",
            year, month, rainfall);
    }
    return 0;
}
    
```

yearly2.c  
6

**Example (cont): YearlyRainfall-1**

```
#define NYEARS 5
#define NMONTHS 12

int lookup(int year, int month)
{
    int table[NYEARS][NMONTHS] =
    {
        {30,40,75,95,130,220,210,185,135,80,40,45},
        {25,25,80,75,115,270,200,165,85,5,10,0},
        {35,45,90,80,100,205,135,140,170,75,60,95},
        {30,40,70,70,90,180,180,210,145,35,85,80},
        {30,35,30,90,150,230,305,295,60,95,80,30}
    };

    if ((0 <= year) && (year < NYEARS) &&
        (0 <= month) && (month < NMONTHS))
    {
        return table[year][month];
    }
    else
    {
        return -1;
    }
}
```

yearly1.c 7

**Example (cont) : YearlyRainfall-1**

```
#define NYEARS 5
#define NMONTHS 12

int lookup(int year, int month)
{
    int table[NYEARS][NMONTHS] =
    {
        {30,40,75,95,130,220,210,185,135,80,40,45},
        {25,25,80,75,115,270,200,165,85,5,10,0},
        {35,45,90,80,100,205,135,140,170,75,60,95},
        {30,40,70,70,90,180,180,210,145,35,85,80},
        {30,35,30,90,150,230,305,295,60,95,80,30}
    };

    if ((0 <= year) && (year < NYEARS) &&
        (0 <= month) && (month < NMONTHS))
    {
        return table[year][month];
    }
    else
    {
        return -1;
    }
}
```

yearly1.c 8

**Example (cont): YearlyRainfall-2**

```
int main()
{
    int year;
    int month;
    int rainfall;

    printf("Enter year and month: ");
    scanf("%d %d", &year, &month);

    rainfall = lookup(year - 1, month - 1);

    if (rainfall < 0)
    {
        printf("Year must be between 1 and %d.\n", NYEARS);
        printf("and month must be between 1 and %d.\n", NMONTHS);
    }
    else
    {
        printf("Rainfall for year %d, month %d is %d mm.\n",
            year, month, rainfall);
    }
    return 0;
}
```

yearly1.c 9

**Example (cont): YearlyRainfall-2**

```
int main()
{
    int year;
    int month;
    int rainfall;

    printf("Enter year and month: ");
    scanf("%d %d", &year, &month);

    rainfall = lookup(year - 1, month - 1);

    if (rainfall < 0)
    {
        printf("Year must be between 1 and %d.\n", NYEARS);
        printf("and month must be between 1 and %d.\n", NMONTHS);
    }
    else
    {
        printf("Rainfall for year %d, month %d is %d mm.\n",
            year, month, rainfall);
    }
    return 0;
}
```

yearly1.c 10

**Example (cont): YearlyRainfall-2**

```
int main()
{
    int year;
    int month;
    int rainfall;

    printf("Enter year and month: ");
    scanf("%d %d", &year, &month);

    rainfall = lookup(year - 1, month - 1);

    if (rainfall < 0)
    {
        printf("Year must be between 1 and %d.\n", NYEARS);
        printf("and month must be between 1 and %d.\n", NMONTHS);
    }
    else
    {
        printf("Rainfall for year %d, month %d is %d mm.\n",
            year, month, rainfall);
    }
    return 0;
}
```

yearly1.c 11

**Passing Two-Dimensional Arrays to Functions**

- In the function definition, the size of the array needs to be specified
- Any changes to array elements within the function affect the "original" array elements

12

Example 2: 2-D Array-1

```
#include <stdio.h>
#define NROWS 3
#define NCOLS 5

void inputEntry(float table[][NCOLS]);
void printTable(float table[NROWS][NCOLS]);

int main()
{
    float table[NROWS][NCOLS] = {{0}};
    printTable(table);

    while (1)
    {
        inputEntry(table);
        printTable(table);
    }
    return 0;
}
```

twod2.c

13

Example 2 (cont): 2-D Array-1

```
#include <stdio.h>
#define NROWS 3
#define NCOLS 5

void inputEntry(float table[][NCOLS]);
void printTable(float table[NROWS][NCOLS]);

int main()
{
    float table[NROWS][NCOLS] = {{0}};

    printTable(table);

    while (1)
    {
        inputEntry(table);
        printTable(table);
    }
    return 0;
}
```

twod2.c

14

Example 2 (cont): 2-D Array-2

```
/* Reads in a location in the table and the value of
one item to be put in the table */
void inputEntry ( float table[NROWS][NCOLS] )
{
    int row, column;
    printf("Enter row and column number: ");
    scanf("%d %d", &row, &column);

    if ((0 <= row && row < NROWS) &&
        (0 <= column && column < NCOLS))
    {
        printf("Enter value: ");
        scanf("%f", &table[row][column]);
    }
    else
    {
        printf("Invalid entry location. No change.\n");
    }
}
```

twod2.c

15

Example 2 (cont): 2-D Array-2

```
/* Reads in a location in the table and the value of
one item to be put in the table */
void inputEntry ( float table[][NCOLS] )
{
    int row, column;
    printf("Enter row and column number: ");
    scanf("%d %d", &row, &column);

    if ((0 <= row && row < NROWS) &&
        (0 <= column && column < NCOLS))
    {
        printf("Enter value: ");
        scanf("%f", &table[row][column]);
    }
    else
    {
        printf("Invalid entry location. No change.\n");
    }
}
```

twod2.c

16

Example 2 (cont): 2-D Array-2

```
/* Reads in a location in the table and the value of
one item to be put in the table */
void inputEntry ( float table[][NCOLS] )
{
    int row, column;
    printf("Enter row and column number: ");
    scanf("%d %d", &row, &column);

    if ((0 <= row && row < NROWS) &&
        (0 <= column && column < NCOLS))
    {
        printf("Enter value: ");
        scanf("%f", &table[row][column]);
    }
    else
    {
        printf("Invalid entry location. No change.\n");
    }
}
```

twod2.c

17

Example 2 (cont): 2-D Array-2

```
/* Reads in a location in the table and the value of
one item to be put in the table */
void inputEntry ( float table[][NCOLS] )
{
    int row, column;
    printf("Enter row and column number: ");
    scanf("%d %d", &row, &column);

    if ((0 <= row && row < NROWS) &&
        (0 <= column && column < NCOLS))
    {
        printf("Enter value: ");
        scanf("%f", &table[row][column]);
    }
    else
    {
        printf("Invalid entry location. No change.\n");
    }
}
```

twod2.c

18

Example 2 (cont): 2-D Array-3

```

/* Prints the table page-by-page, and each page
row-by-row */
void printTable ( float table[NROWS][NCOLS] )
{
    int row, column;
    for (row=0; row < NROWS; row++)
    {
        for (column=0; column < NCOLS; column++)
        {
            printf("%10.2f", table[row][column]);
        }
        printf("\n");
    }
}
    
```

twod2.c

Example 2 (cont): 2-D Array-3

```

/* Prints the table page-by-page, and each page
row-by-row */
void printTable ( float table[NROWS][NCOLS] )
{
    int row, column;
    for (row=0; row < NROWS; row++)
    {
        for (column=0; column < NCOLS; column++)
        {
            printf("%10.2f", table[row][column]);
        }
        printf("\n");
    }
}
    
```

twod2.c

Compiler Problems with Floats

- Problem: You may encounter the following run-time error message with Borland C/C++ and Turbo C/C++:

*scanf : floating point formats not linked  
Abnormal program termination*

Compiler Problems with Floats (cont)

Explanation:

From Section Q:03.04 in <http://www.faqs.org/faqs/msdos-programmer-faq/part2/>

"floating point formats not linked" is a Borland run-time error (Borland C or C++, Turbo C or C++). Borland's compilers try to be smart and not link in the floating-point (f-p) library unless you need it. Alas, they all get the decision wrong. One common case is where you don't call any f-p functions, but you have %f or other f-p formats in scanf() or printf() calls. The cure is to call an f-p function, or at least force one to be present in the link.

Compiler Problems with Floats (cont)

- Solution:  
Define the following function somewhere in your source file, but do not call it:

```

void dummyFunction ( void )
{
    float dummyFloat;

    scanf("%f", &dummyFloat);
    printf("%f", dummyFloat);
}
    
```

Reading

- King
  - Chapter 8, Chapter 12 (12.2-12.4)
- Deitel and Deitel
  - Chapter 6 (6.5 to 6.9)