CSE 142
Computer Programming I

Arrays
Structures
Multidimensional Arrays
Arrays of Structures
Structures of Arrays
Arrays of Structures of Arrays of…

© 2000 UW CSE

Overview

When do I use an array, and when do I use a structure?

General Rule

Array: Holds multiple instances of one logical value

Examples:

- Amount of rain each day during one week
- The MT2 grade of each student
- Number of shares of MSFT sold each trading day during 2000
- Number of shares sold on Dec. 12, 2000 of each company listed on the NYSE
- Name of each driver in the Duralube 500

Array Examples

Examples:

- Amount of rain each day during one week
- The MT2 grade of each student
- Number of shares of MSFT sold each trading day during 2000
- Number of shares sold on Dec. 12, 2000 of each company listed on the NYSE
- Name of each driver in the Duralube 500

0	35
1	22
2	40
3	38
4	25
5	28
…	…
499	33

Structures: General Rule

Structure: Holds multiple characteristics of one logical instance

Examples:

- {Amount of rain, average temperature, average relative humidity} of one day
- {Name, student ID, MT2 grade} of one student
- {Total #shares traded, high price, low price, avg. price} of MSFT on one trading day
- {Company name, stock symbol, corporate address} of one NYSE-listed company
- {Driver name, primary sponsor name, age} of one driver

Other Examples

- The number of salmon counted in the Cedar River each day of 1999
- The names of all the winners of the Nobel Prize in Literature
- The number of Ford Focuses on each Ford dealer's lot in Western Washington
- The number of Ford Focuses of each color on each Ford dealer's lot in Western Washington

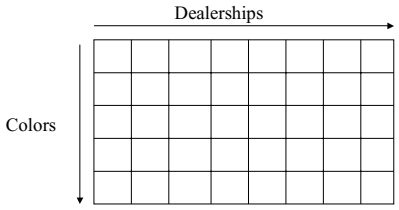
The number of Ford Focuses of each color on each Ford dealer's lot in Western Washington

Array: Holds multiple instances of one logical value

Structure: Holds multiple characteristics of one logical instance

R-7

The number of Ford Focuses of each color on each Ford dealer's lot in Western Washington



Two indexing values: dealership and color

R-8

The number of Ford Focuses of each color on each Ford dealer's lot in Western Washington

	A	B	C	D	E	….		Z
Red	0	2	0	12	1	…		4
Silver	3	8	1	2	0			0
Green								
White		…						
Black	0	1	1	3	5	…		9

Two "selection criteria": dealership and color
One logical value: number of cars on lot

R-9

Multi-Dimensional Arrays

Each dimension corresponds to some "selection criterion"
- Color
- Dealership

All array entries tell you "the same logical value" for different selection criteria values
- Number of cars

R-10

Multi-Dimensional Arrays

Other examples:
- The score of each student on each assignment in CSE 142 during Winter 2001
- The score of each student on each assignment in CSE142 (ever)
- The number of copies of each book at each branch library of the Seattle Public Libraries
- The number of people arrested on each felony count during each hour between midnight and 4:00am of each day of the week of Mardi Gras

R-11

Multi-Dimensional Arrays in C

There's only one "logical value" being stored -> all elements of the array (must) have the same type.

The "selection criteria" have to be expressed as integers:
- If my array is "# of cars vs. (color,dealership)", I need to map colors to integers and dealership to integers

R-12

The "selection criteria" have to be expressed as integers

	0	1	2	3	4	….		25
(Red) 0	0	2	0	12	1	…		4
(Silver) 1	3	8	1	2	0			0
(Green) 2								
(White) 3		…						
(Black) 4	0	1	1	3	5	…		9

Two "selection criteria": dealership and color
One logical value: number of cars on lot

R-13

2-Dimensional Arrays

Example: scores for 7 students on 4 homeworks

score hw	0	1	2	3
student 0	22	15	25	25
student 1	12	12	25	20
student 2	5	17	25	24
student 3	15	19	25	13
student 4	2	0	25	25
student 5	25	22	24	21
student 6	8	4	25	12

C expressions:

score[0][0] is 22

score[6][3] is 12

2*score[3][0] is 30

R-14

Declaring a 2-D Array

```
#define MAX_STUDENTS   80
#define MAX_HWS    6
...
int score [MAX_STUDENTS] [MAX_HWS] ;
```

R-15

2-D Arrays: Terminology

type name[#rows][#columns]

int score[80][6];

score is a *two-dimensional array of int* of size 80 by 6

score[0][0], score[0][1], ... , score[79][5] are the elements of the array

R-16

Bookkeeping

As with 1-D arrays, often we only use part of the space available in a 2-D array

Declared size of the array specifies its *maximum capacity.*

The *current size* (# of rows and columns currently in use) needs to be kept track of in separate variables

R-17

Reading in Data

Problem: Read in data for student assignments

Input data format: The number of students, then the number of assignments, followed by the data per student

A nested loop is the right program structure for reading in the data details

```
int score [MAX_STUDENTS] [MAX_HWS] ;
int nstudents, nhws, i, j ;
```

R-18

Reading a 2-D Array: Code

```
/* Read the number of students and assignments,
   then loop to read detailed data */

scanf ("%d %d", &nstudents, &nhws) ;
if (nstudents <= MAX_STUDENTS &&
      nhws <= MAX_HWS) {
    for ( i = 0 ;  i < nstudents ;  i = i + 1 )
      for ( j = 0 ;  j < nhws ;  j = j + 1 )
        scanf("%d", &score [ i ] [ j ]) ;
}
```

R-19

Part of the array is unused; which part?

Array Input Trace

Input: 7 4 0 1 2 3 4 5 6 7 8 9 ...

score	j= 0	1	2	3	4	5 ...
i=0	0	1	2	3	?	? ...
i=1	4	5	6	7	?	? ...
i=2	8	9	...			
...			...			
i=6	...					
i=7	?	?	?	?	...	

R-20

Printing a 2-D Array

```
if (nstudents <= MAX_STUDENTS &&
      nhws <= MAX_HWS) {
    for ( i = 0 ;  i < nstudents ;  i = i + 1 ) {
      for ( j = 0 ;  j < nhws ;  j = j + 1 ) {
        printf("%d", score [ i ] [ j ]) ;
      }
      printf("\n") ;
    }
}
```

R-21

2-D Arrays as Parameters

Same as 1-D arrays (almost):

- Individual array elements can be either value or pointer parameters
- Entire arrays are always passed as pointer parameters - never copied
- Don't use & and * with entire array parameters

Difference:

No empty brackets [ ] in formal parameters

Actually, [ ] allowed sometimes; we won't use in this course

R-22

2-D Array As Parameter

A function to read into array grades the grade information for the given number of students and assignments

```
void read_2D ( int grades[MAX_STUDENTS] [MAX_HWS],
                 int nstudents, int nhws)
{...
```

R-23

2-D Array As Parameter

```
/* Read into array grades the grade information for */
/* the given number of students and assignments */

void read_2D ( int grades[MAX_STUDENTS] [MAX_HWS] ,
                 int nstudents, int nhws)
{
   int i, j ;
   for ( i = 0 ;  i < nstudents ;  i = i + 1 )
     for ( j = 0 ;  j < nhws ;  j = j + 1 )
       scanf("%d", &grades[ i ][ j ]) ;
}
```

R-24

Array Function Arguments

```
int main(void) {
   int scores [MAX_STUDENTS] [MAX_HWS] ;
   int nstudents, nhws ;

   scanf ("%d %d", &nstudents, &nhws) ;
   if ( nstudents <= MAX_STUDENTS &&
        nhws <= MAX_HWS)
           read_2D (scores, nstudents, nhws) ;
    ...
}
```

no &

R-25

Arrays of Structures

Structure: Holds multiple characteristics of one logical instance

Array: Holds multiple instances of one logical value

Examples:

- {Amount of rain, average temperature, average relative humidity} on each day of 2000
- {Name, student ID, MT2 grade} of each 142 student
- {Total #shares traded, high price, low price, avg. price} of MSFT on each trading day of 2000
- {Company name, stock symbol, corporate address} of each NYSE-listed company

R-26

•{Amount of rain, average temperature, average relative humidity} on each day of 2000

```
#define MAXDAYS        366

typedef structure {
                double rainFall;
                double avgTemperature;
                double relativeHumidity;
} ClimateData;

ClimateData      seattle2000[MAXDAYS];

…

seattle2000[0].rainFall = 1.1;
seattle2000[365].avgTemperature = 45.2;
```

R-27

•{Amount of rain, average temperature, average relative humidity} on each day of each year of the 1900's

```
#define MAXDAYS        366
#define MAXYEARS       100
typedef structure {
                double rainFall;
                double avgTemperature;
                double relativeHumidity;
} ClimateData;

ClimateData      seattle[MAXYEARS][MAXDAYS];

…

seattle [0][0].rainFall = 1.1;                  // 1/1/1900
seattle[99][365].avgTemperature = 45.2;         // 12/31/1999
```

R-28

•{Amount of rain, average temperature, average relative humidity} on each day of each year of the 1900's

```
#define MAXDAYS        366
#define MAXYEARS       100
typedef structure {
                double rainFall;
                double avgTemperature;
                double relativeHumidity;
} ClimateData;

ClimateData      seattle [MAXDAYS] [MAXYEARS];

…

seattle [0][0].rainFall = 1.1;                  // 1/1/1900
seattle[365][99].avgTemperature = 45.2;         // 12/31/1999
```

R-29

•{Amount of rain, average temperature, average relative humidity} during each hour of each day of each year of the 1900's

```
#define MAXHOURS       24
#define MAXDAYS        366
#define MAXYEARS       100
typedef structure {
                double rainFall;
                double avgTemperature;
                double relativeHumidity;
} ClimateData;

ClimateData      seattle [MAXHOURS][MAXDAYS] [MAXYEARS];

…

seattle [0][0][0].rainFall = 1.1;               // midnight to 1:00am 1/1/1900
Seattle[23][365][99].avgTemperature = 45.2; // 11:00pm to midnight 12/31/1999
```

R-30

Structs Containing Arrays

A student record has:

- Student ID number
- Grade on each assignment

```
#define MAXASSIGNMENTS        10

typedef struct {
    int          ID;
    double       grade[MAXASSIGNMENTS];
} StudentRecord;
…
StudentRecord JZ;
JZ.grade[0] = 0;        // no points for JZ on first graded assignment
```

R-31

Arrays of Structs with Arrays…

```
#define MAXASSIGNMENTS        10
#define MAXSTUDENTS           600
typedef struct {
    int          ID;
    double       grade[MAXASSIGNMENTS];
} StudentRecord;
…
StudentRecord allStudents[MAXSTUDENTS];
allStudents[20].grade[0] = 44;
scanf("%lf", &allStudents[33].grade[4]);
…
```

R-32