UCSC Silicon Valley Extension

Advanced C Programming

More on Structures

Instructor: Radhika Grover

Overview

- Creating arrays of structures
- Return a structure from a function
- Sorting an array of structures

Arrays of structures

- The following arrays can hold only one type of value

  int array[10]; **double** array1[200];

- Array of structures can be used to hold different types of values.

Example:

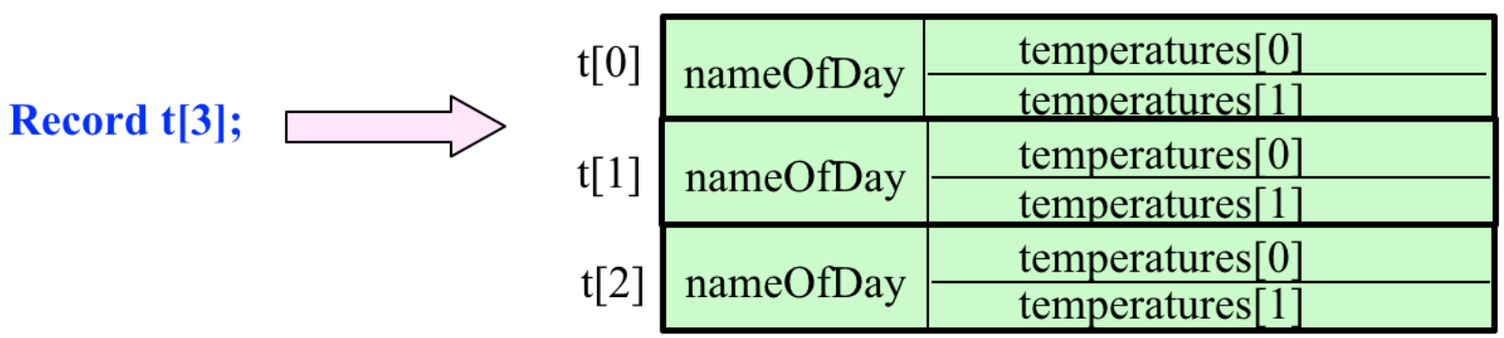
```
Record r[100]; /* Each element in this array is a structure of type Record */
student s[500]; /* Each element in this array is a structure of type student */
```

Arrays of structures

Example: We have the following structure called Record defined below:

```
typedef struct {
    char nameOfDay[10];
    int temperatures[2];
} Record;
```

We can declare an array called *t* of type *Record* that can hold 3 records as follows:

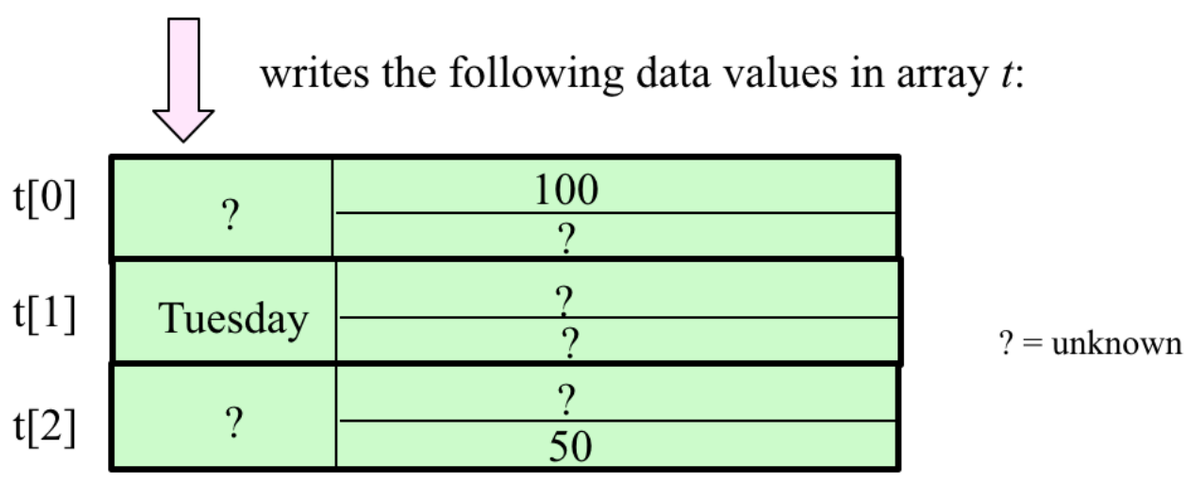


Modifying data members of a structure

- We can access data members of array of structures *t* using the dot operator:
  array_name[index] **.** member_name

Modifying the data members of t[3]:

```
t[0].temperatures[0] = 100;
strncpy(t[1].nameOfDay, “Tuesday”, 9);
t[1].nameOfDay[7] = ‘\0’;
t[2].temperatures[1] = 50;
```



Example 3: Array of structures

- Rewrite the program in Structures2/example2.c so that we can store data for *five days* in an array of the following structure.

```
typedef struct {
        char nameOfDay[10];
        int temperatures[2];  /* pointer to array storing temperatures at 2 different times */
} Record;
```

Use the functions *change* and *print* from example 2, without modifying them.

(Hint: only the main function will change)

Example 3: Array of structures

```
// Structures3/example3.c
int main(void) {
        int i;
        Record r1[5] = { { "none", { -1, -1 } }};

        /* Call function change */
        printf("\n\n Calling function change to read input \n");
        for (i = 0; i < 5; i++)
                change(&r1[i]);  /* Note: we are passing the address of  “r1[i]” */
        for (i = 0; i < 5; i++)
                print(r1[i]);
        return 0;
}
```

Compile and run this program with the declarations and functions in Example 2. Do not include function *noChange*.

Example 4: Returning a structure from a function

```
// Structures4/example4.c
/* This function called readRecord reads data from the keyboard and returns it in a Record structure. */
Record readRecord(void) {/* returning structure by value */
	Record r;
	int i;
	printf("Enter day of week:");
	scanf("%9s", r.nameOfDay);
	for( i = 0; i < 2; i++ ) {
		printf("Enter temperature for hour %d:", i);
		scanf("%d", &r.temperatures[i] );
	}
	return r;
}
```

Why would it be incorrect to return a pointer to the Record r from function readRecord()?

Example 4

- The main program for function on the previous slide:

```
// Structures4/example4.c
int main(void) {
    Record r1 = { "none", { -1, -1 } };

    /* Call function change */
    printf("\n\n Calling function change to read input \n");
    r1 = readRecord();  /* Note: the structure returned by function change is
copied  into r1*/
    print(r1);     /* This function was defined earlier to print a Record */
    return 0;
}
```

Exercise 5

The following structure can be used to represent information for comets.

```
struct Comet {
        char name[50];
        int yearOfFirstSighting;
        float period;
};
```

Write a program to read this information from a file called “comets.txt” into an array of the above structure. Write a function sort() to sort the elements of the array of structure so that it is ordered (ascending) by the field yearOfFirstSighting. Write a function to print out the array of structure.

Exercise 5 solution : header file

```
// Structure5/example5.c
#ifndef EXAMPLE5_H_
#define EXAMPLE5_H_

#include <stdio.h>
#include <string.h>

#define FILENAME "comets.txt"
#define MAX_SIZE 100

typedef struct  {
            char name[50];
            int yearOfFirstSighting;
            float period;
      } Comet;

void printArray(Comet arr[], int size); /* print out the contents of array arr */
void sort(Comet arr[], int size);          /* sort the array a by field yearOfFirstSighting */
void swap(Comet *c1_ptr, Comet *c2_ptr);  /* swap the contents of Comet structures                    pointed
to by c1_ptr and c2_ptr */

#endif /* EXAMPLE5_H_ */
```


Exercise 5 solution : sort()

```
// Structure5/example5.c
void sort(Comet a[], int size) {
        int j, i, min;
        for ( j = 0; j < size; j++ ) {
                min = j;
                /* find the smallest number from a[j+1] to a[j] */
                for ( i = j+1; i < size; i++) {
                        if( a[i].yearOfFirstSighting < a[min].yearOfFirstSighting )
                                min = i;
                }
                /* swap the structures at index j & min*/
                swap(&a[j], &a[min]);
        }

}
```

Exercise 5 solution : swap()

```
// Structure5/example5.c
void swap(Comet *c1_ptr, Comet *c2_ptr) {
	Comet temp;
	int len = sizeof(temp.name);
	// copy c1_ptr's pointee into temp
	strncpy(temp.name, c1_ptr->name, len- 1);
	temp.name[len - 1] = '\0';
	temp.yearOfFirstSighting = c1_ptr->yearOfFirstSighting;
	temp.period = c1_ptr->period;

	// copy c2_ptr's pointee into c1_ptr's pointee
	strncpy(c1_ptr->name, c2_ptr->name, len - 1);
	c1_ptr->name[len - 1] = '\0';
	c1_ptr->yearOfFirstSighting = c2_ptr->yearOfFirstSighting;
	c1_ptr->period = c2_ptr->period;

	// copy temp into c2_ptr's pointee
	strncpy(c2_ptr->name, temp.name, len - 1);
	c1_ptr->name[len - 1] = '\0';
	c2_ptr->yearOfFirstSighting = temp.yearOfFirstSighting;
	c2_ptr->period = temp.period;
}
```

Exercise 5 solution : printArray()

```
// Structure5/example5.c
void printArray(Comet arr[], int size) {
	int j;
	for (j = 0; j < size; j++) {
		printf("Comet name: %s \t", arr[j].name);
		printf("Year of first sighting %d \t", arr[j].yearOfFirstSighting);
		printf("period: %f \n", arr[j].period);
	}
}
```

Exercise 5 solution : main

```
// Structure5/example5.c
int main(void) {
	Comet r[MAX_SIZE];
	FILE *fp;
	int i = 0, val;
	fp = fopen(FILENAME, "r");
	if(fp == NULL) {
		printf("Error opening file \n");
		return 1;
	}
	/* Read data from file into structure r and print it to the screen */
	while( (val = fscanf(fp, "%49s %d %f", r[i].name, &r[i].yearOfFirstSighting, &r[i].period)) != EOF ) {
		printf("%s %d %f  \n", r[i].name, r[i].yearOfFirstSighting, r[i].period);
		i++;
	}
	/* sort the array, i is the number of elements in the array */
	sort(r, i);
	/* print the sorted array */
	printArray(r, i);
	fclose(fp); /* close the file pointer */
	return 0;
}
```

