CSE / ENGR 142 Programming I

Arrays of Structures

Structures and Arrays

- A *struct* represents a single record
- Typically, *struct*s are used to deal with collections of such records
- Examples: student records, employee records, customer records, parts records
- In each case we will have multiple instances of the record (struct) type

Arrays of structs are the natural way to do this

Arrays of *struct*s

Each declaration below declares an array, where each array element is a structure:

```
point corner_points[10];

time meeting_times[MAX_MEETINGS];

student_record cse_142[MAX_STUDENTS];
```

Using arrays of *struct*s is a natural extension of principles we're already learned.

Using Arrays of *struct*s

- We access a field of a *struct* in an array by specifying the array element and then the field:

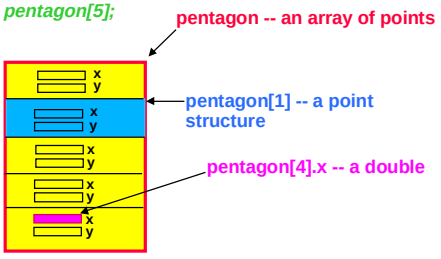
```
cse_142[i].name

corner_points[j+1].x

meeting_times[4].hours
```


Naming in *struct* Arrays

```
point pentagon[5];
```



Using Arrays of *struct*s

```
student_record  class[MAX_STUDENTS] ;
...
for ( i = 0 ;  i < nstudents ;  i = i + 1 )
{
   scanf("%d %d", &class[i].hw, &class[i].exams) ;
   class[i].grade =
      (double) (class[i].hw + class[i].exams) / 50.0 ;
}
```

struct Array Elements as Parameters

```
void draw_line ( point p1, point p2) {.....}
...
point pentagon[5] ;
...
for ( i = 0 ;  i < 4 ;  i = i + 1)
    draw_line (pentagon[i], pentagon[i+1]) ;
draw_line (pentagon[4], pentagon[0]) ;
```


*struct*s and *struct* Arrays as Parameters

- A single *struct* is passed by value
  - all of its components are copied from the argument (actual parameter) to initialize the (formal) parameter.

```
point set_midpt (point a; point b) {...}
int main (void) {
    point p1, p2, m;          /* declare 3 points */
    ...
    m = set_midpt ( p1, p2);
}
```


Passing Arrays of *struct*s

- An array of *struct*s is an array.
- When any array is an argument (actual parameter), it is passed by reference (not copied). [As for any array]
  - The parameter is an alias of the actual array argument

```
int avg (student_rec class_db[MAX_N]) {...}
int  main (void) {
    student_rec cse_142[MAX_N];
    int average;
    ....
    average = avg ( cse_142 ); /* by reference */
}
```


Sorting Arrays of *struct*s

David	Kathryn	Sarah	Phil	Casey
920915	901028	900317	920914	910607
2.9	4.0	3.9	2.8	3.6

Phil	David	Casey	Sarah	Kathryn
920914	920915	910607	900317	901028
2.8	2.9	3.6	3.9	4.0

```
typedef struct  {
    char      name [MAX_NAME + 1] ;
    int       id ;
    double    score ;
} student_record ;
```


Insertion Sort for *student_record*

```
void sort (student_record a[ ],  int size)
{
    int j ;
    for ( j = 1 ;  j < size ;  j = j + 1 )
      insert (a, j) ;
}
```


Insert for sorting records

```
void insert ( student_record a[ ] , int j )
{   int i;
    student_record temp ;
    temp = a[j] ;
    for ( i = j;
          i > 0  && a[i-1].score > temp.score;
          i = i - 1 ) {
      a[i] = a[i-1];
    }
    a[i] = temp ;
}
```

Alphabetical Order

David	Casey	Sarah	Phil	Kathryn
920915	910607	900317	920914	901028
2.9	3.6	3.9	2.8	4.0

Casey	David	Kathryn	Phil	Sarah
910607	920915	901028	920914	900317
3.6	2.9	4.0	2.8	3.9

```
typedef struct  {
     char      name[MAX_NAME + 1];
     int       id;
     double    score;
} student_record;
```


String Comparison: *strcmp*

"Alice" is less than "Bob"
"Dave" is less than "David"
"Rob" is less than "Robert"

```
#include <string.h>
int  strcmp (char str1[ ], char str2[ ])
```

returns negative integer if str1 is less than str2
0 if str1 equals str2
positive integer if str1 is greater than str2

Insert for Alphabetic Sorting

```
void insert ( student_record a[ ], int j )
{   int i;
    student_record temp ;

    temp = a[j] ;
    for ( i = j ;
          i > 0  && strcmp (a[i-1].name, temp.name) > 0 ;
          i = i - 1 ){
       a[i] = a[i-1];
    }
    a[i] = temp ;
}
```


Type Quiz

```
student_record a [MAX_STUDENTS];
```

a
a[0]
a[5].name
a[4].id
&a[6].score
a[2].name[1]

Data Structures: What If...

- **...you wanted to keep information about one song on the computer.**
  - **What pieces of data would you want?**
  - **How would you organize them?**
  - **How would it look in C?**
- **And then…**
  - **What if you wanted information about an entire CD of songs?**
  - **And then… how about a whole collection of CD's?**