# CSE 142 Programming I

## Arrays of Structures

## Structures and Arrays

- A *struct* represents a single record
- Typically, computer applications have 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 one record (struct) type

*Arrays of structs are the natural way to do this*

## Recall These *struct* Examples

```
#define MAX_NAME  40
typedef struct {
    char name [MAX_NAME+1] ;
    int id ;
    double  score ;
} student_record ;
```

```
typedef struct {
    int hours, minutes ;
    double  seconds ;
}  time ;

typedef struct {
      double  x, y ;
} point ;
```

## Arrays of *structs*

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 *structs* is a natural extension of principles we're already learned.

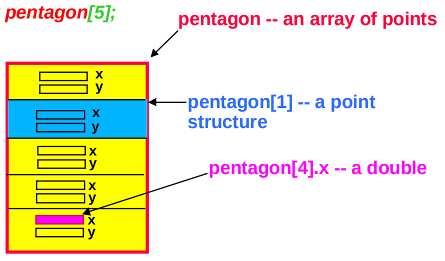
## Using Arrays of *structs*

- 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];
```


pentagon -- an array of points
x
y
pentagon[1] -- a point structure
pentagon[4].x -- a double

## Using Arrays of *structs*

```
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]) ;
```

## Review: *structs* 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 midpoint (point a; point b) {...}

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

## Passing Arrays of *structs*

- An array of *structs* is an array.
- When *any* array is an argument (actual parameter), it is passed by reference (not copied).]
  - 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 *structs*

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

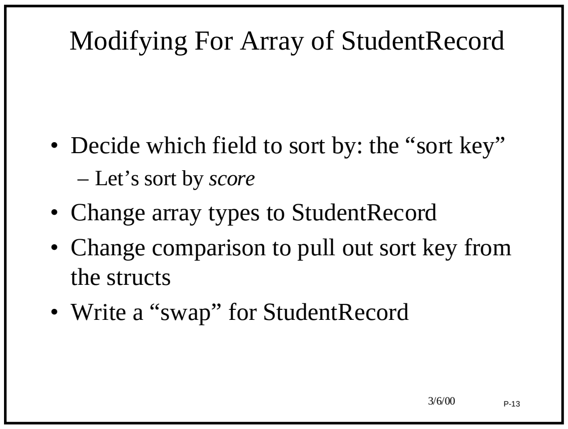
| | | | | |
|---|---|---|---|---|
| Phil 920914 2.8 | David 920915 2.9 | Casey 910607 3.6 | Sarah 900317 3.9 | Kathryn 901028 4.0 |

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

## Review: Selection Sort for Array of ints

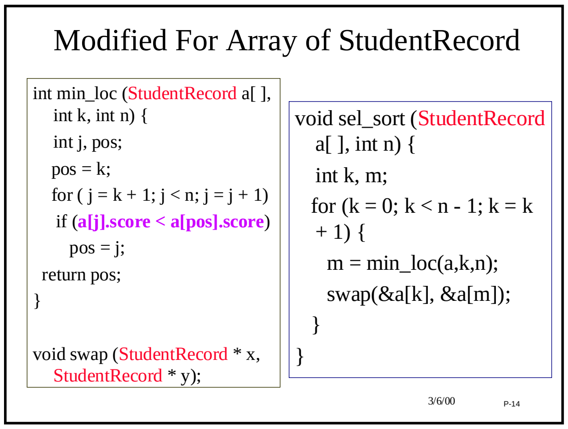
```
int min_loc (int a[ ], int k, int n)
    {
    int j, pos;
    pos = k;
    for ( j = k + 1; j < n; j = j + 1)
      if (a[j] < a[pos])
        pos = j;
  return pos;
}

void swap (int * x, int * y);
```

```
void sel_sort (int a[ ], int n)
  {
  int k, m;
  for (k = 0; k < n - 1; k = k
   + 1) {
     m = min_loc(a,k,n);
     swap(&a[k], &a[m]);
   }
}
```



## Modifying For Array of StudentRecord

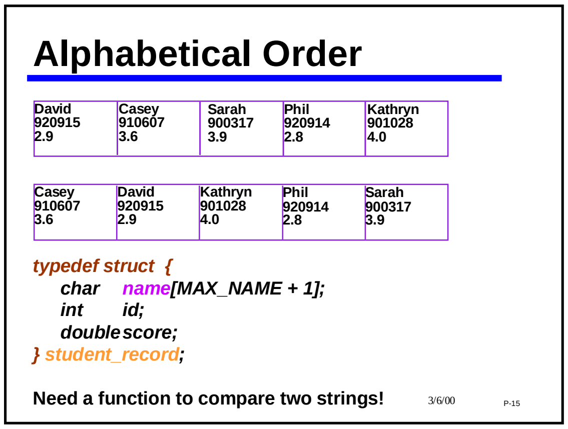
- Decide which field to sort by: the “sort key”
  - Let’s sort by *score*
- Change array types to StudentRecord
- Change comparison to pull out sort key from the structs
- Write a “swap” for StudentRecord



## Modified For Array of StudentRecord

```
int min_loc (StudentRecord a[ ],
    int k, int n) {
    int j, pos;
    pos = k;
    for ( j = k + 1; j < n; j = j + 1)
     if (a[j].score < a[pos].score)
       pos = j;
  return pos;
}

void swap (StudentRecord * x,
    StudentRecord * y);
```

```
void sel_sort (StudentRecord
    a[ ], int n) {
    int k, m;
   for (k = 0; k < n - 1; k = k
    + 1) {
      m = min_loc(a,k,n);
      swap(&a[k], &a[m]);
    }
}
```



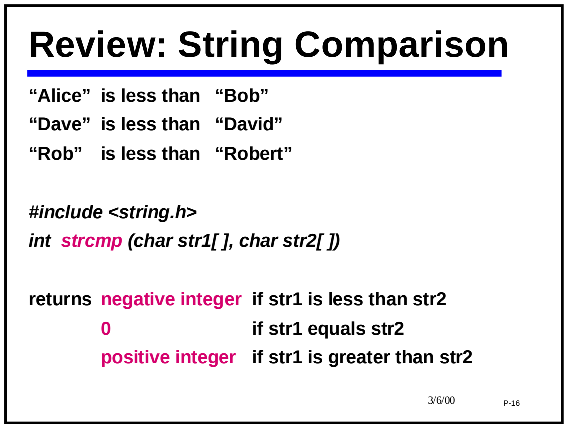
## Alphabetical Order

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

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

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

Need a function to compare two strings!

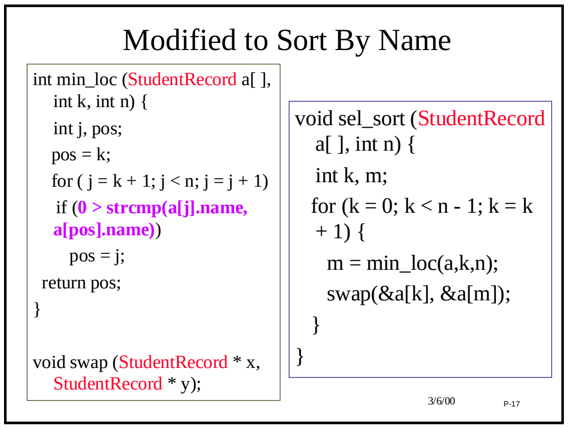


## Review: String Comparison

| | | |
|---|---|---|
| “Alice” | is less than | “Bob” |
| “Dave” | is less than | “David” |
| “Rob” | is less than | “Robert” |

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

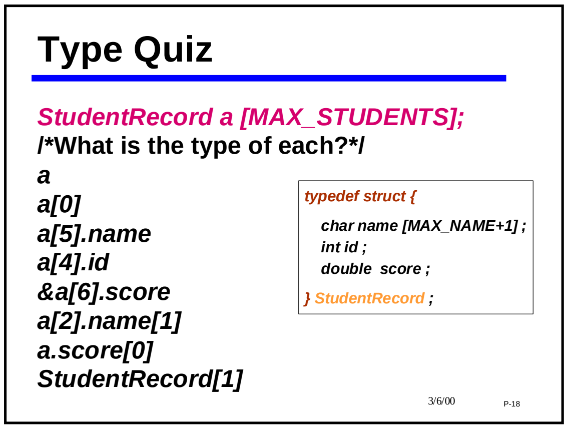
| | | |
|---|---|---|
| returns | negative integer | if str1 is less than str2 |
| | 0 | if str1 equals str2 |
| | positive integer | if str1 is greater than str2 |



## Modified to Sort By Name

```
int min_loc (StudentRecord a[ ],
    int k, int n) {
    int j, pos;
    pos = k;
    for ( j = k + 1; j < n; j = j + 1)
     if (0 > strcmp(a[j].name,
    a[pos].name))
       pos = j;
  return pos;
}

void swap (StudentRecord * x,
    StudentRecord * y);
```

```
void sel_sort (StudentRecord
    a[ ], int n) {
    int k, m;
   for (k = 0; k < n - 1; k = k
    + 1) {
      m = min_loc(a,k,n);
      swap(&a[k], &a[m]);
    }
}
```



## Type Quiz

```
StudentRecord a [MAX_STUDENTS];
/*What is the type of each?*/
a
a[0]
a[5].name
a[4].id
&a[6].score
a[2].name[1]
a.score[0]
StudentRecord[1]
```

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

## 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?

- Not covered Winter 00: sorting using Insertion Sort (following slides)

## Insertion Sort for *student_record*

```
/* sort student records a[0..size-1] in */
/* ascending order by score */
void sort (student_record a[ ],  int size)
{
    int j ;
    for ( j = 1 ;  j < size ;  j = j + 1 )
      insert (a, j) ;
}
```

## *Insert* for sorting records

```
/* given that a[0..j-1] is sorted, move a[j] to the correct */
/* location so that that a[0..j] is sorted by score */
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 ;
}
```

## *Insert* for Alphabetic Sorting

```
/* given that a[0..j-1] is sorted, move a[j] to the correct */
/* location so that that a[0..j] is sorted by name */
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 ;
}
```