

Structures

CS 241

Data Organization using C

Instructor: **Joel Castellanos**  
e-mail: [joel@unm.edu](mailto:joel@unm.edu)  
Web: <http://cs.unm.edu/~joel/>  
Office: Farris Engineering  
Center (FEC) room 321  
Lab Instructor: **Dongye Meng**  
e-mail: [dymeng@cs.unm.edu](mailto:dymeng@cs.unm.edu)



3/10/2009

## Textbook Section 6.1: Basics of Structure

//**x** and **y** are members of the structure point.

```
struct point {int x; int y;};
```

//A structure does not reserve storage.

//It only defines the type of storage.

```
struct point pt; //This reserves storage.
```

// Alternate syntax defining and instantiating a structure.

```
struct point  
{ int x;  
  int y;  
} pt;
```

2

## Accessing the Members of a Structure

```
struct point
{ int x;
  int y;
} pt;
```

//Assignment to the members of a structure.

```
pt.x = 5;
pt.y = 8;
```

//Initializing a structure when it is instantiated.

```
struct point maxpt = {320, 200};
```

3

## CS-241 Coding Standard

```
struct point
{ int x;
  int y;
};
struct point pt;
```

```
struct point
{ int x;
  int y;
} pt;
```

```
struct point {int x; int y;} pt;
```



```
struct point {
    int x; int y;
} pt;
```

4

## Section 6.2: Structures and Functions

```
struct point {int x; int y;};
struct point makepoint(int x, int y)
{ //reuse of the variable names x and y is good.
  struct point temp;
  temp.x = x;
  temp.y = y;
  return temp;
}

int main()
{ struct point p1 = makepoint(5,7);
  printf("p1=(%d, %d)\n", p1.x, p1.y);
}
```

Unlike a Java class,  
`temp` is returned by  
value.

5

## Passing a Structures as an Argument

```
struct point {int x; int y;};
void incrementPoint(struct point p)
{ p.x++;
  p.y++;
}

int main()
{ struct point p1 = {4, 7};
  incrementPoint(p1);
  printf("p1=(%d, %d)\n", p1.x, p1.y);
}
```

output: `p1=(4, 7)`

6

## Quiz: Structures and Functions

```
struct point {int x; int y;};
struct point incPoint(struct point p)
{ p.x++; p.y++;
  return p;
}
int main()
{ struct point p1 = {12, 3};
  struct point p2 = incPoint(p1);
  printf("p1=(%d, %d) p2=(%d, %d)\n",
        p1.x, p1.y, p2.x, p2.y);
}
```

- a) p1=(12, 3) p2=(13, 4)      b) p1=(12, 3) p2=(12, 3)  
c) p1=(13, 4) p2=(13, 4)      d) p1=(13, 4) p2=(12, 3)  
e) The value returned into p2 was stored on the stack in incPoint. Therefore, the value in p2 is unpredictable!

7

## Section 6.4: Pointers to Structures

```
struct point {int x; int y;};
void incrementPoint(struct point *p)
{ (*p).x++; // . has higher precedence than *
           // *p.x++; is a syntax error.
  // dereferencing a pointer, then accessing a member is so
  // common that it is given a special notation.
  p->y++;
}
int main()
{ struct point p1 = {4, 7};
  incrementPoint(&p1);
  printf("p1=(%d, %d)\n", p1.x, p1.y);
}
```

output: p1=(5, 8)

8

## Quiz: Pointers to Structures

```
struct point {int x; int y;};
void incrementPoint(struct point *p)
{ (*p).x += 2;
  p->y += 2;
}
int main()
{ struct point p1 = {7, 7};
  incrementPoint(&p1);
  printf("p1=(%d, %d)\n", p1.x, p1.y);
}
```

- |              |              |
|--------------|--------------|
| a) p1=(7, 7) | b) p1=(7, 9) |
| c) p1=(9, 9) | d) p1=(9, 7) |
| e) p1 = 14   |              |

9

## Section 6.3: Arrays of Structures

```
char *keyword[NUM_KEYWORDS];
int keycount[NUM_KEYWORDS];
```

```
struct keyStructure
{ char *word;
  int count;
} key[NUM_KEYWORDS];
```

```
key[0].word = "if"; key[0].count = 0;
key[1].word = "for"; key[1].count = 0;
key[2].word = "char"; key[2].count = 0;
key[3].word = "int"; key[3].count = 0;
```

10

## gcc Compile Error

```
1. #include <stdio.h>
2. #define NUM_KEYWORDS 32;
3. int main()
4. { struct keyStructure
5.   { char *word;
6.     int count;
7.   } key[NUM_KEYWORDS];
8.   ...
```

```
foo.c: In function 'main':
foo.c:7: error: expected `]' before `;' token
```

11

## Linked List Of Quotations

```
#define MAX_QUOTES 2400
struct listRecord
{ int author; //index into heap.data[ ]
  int quote; //index into heap.data[ ]
  int size; //size of author + quote data, including '\0's
  int next; //index into list.rec[ ]
  int previous; //index into list.rec[ ]
};
struct linkedListOfQuotations
{ int start; //index into list.rec[ ]
  int free; //index into list.rec[ ]
  struct listRecord rec[MAX_QUOTES];
} list;
```

12

## Initialize Linked List Of Quotations

```
void initializeLinkedListOfRecords()
{ list.free = 0;
  list.start = -1;

  int i;
  for (i=0; i<MAX_QUOTES; i++)
  { list.rec[i].next = i+1;
  }
  list.rec[MAX_QUOTES-1].next = -1;
}
```

13

## Quiz:

```
#define MAX_QUOTES 7
void initializeLinkedListOfRecords()
{ list.free = 0;
  list.start = -1;
  int i;
  for (i=0; i<MAX_QUOTES; i++)
  { list.rec[i].next = i+1;
  }
  list.rec[MAX_QUOTES-1].next = -1;
  int i=list.free;
  while (i>=0)
  { i=list.rec[i].next;
    printf("%d ", i);
  }
}
```

- a) 0 1 2 3 4 5 6
- b) 1 2 3 4 5 6
- c) 1 2 3 4 5 6 7
- d) 1 2 3 4 5 6 -1
- e) 2 3 4 5 6 -1

14

## Success with Lab 4



- Start with Lab 3:
- Change the doubly linked parallel arrays to the Lab 4 `struct` structure, but leave the data array as it (do not use the heap `struct`). Get this working on lab 3 test files.
- Add in the heap structure, but only use the `heap.data[ ]` part. Get this working on lab 3 test files.
- Add `:print:quotelist`, `:print:data:idx`, and `:echo`. This will get you through `quoteDB-1.txt`
- Add `:query:author:author substring`, and `:query:quote:quotation substring`. This will get you through `quoteDB-2.txt`, and `quoteDB-Hamlet.txt`

15

## `removeHeapRecord()`: Do it the Easy Way First

When you need to merge two heap records, the new merged record may not be in the correct sorted location. the *easiest* way is:

- Always unlink the merged heap record from the heap list list
- Walk through list from `heap.start` to find the insertion point.

```
void unlinkHeapRecord(int i) {...}
int deleteHeapRecord(int dataIdx, int size)
{ ...
  unlinkHeapRecord(newIdx);
  ...
}
```

16

## Read Buffer – easy way first

1. Create a big buffer and do not worry about it.
2. Do not worry about maintaining the biggest record thing - it will only change when you near full.

17

## Quiz: Pointers to Structures

```
#include <stdio.h>
#include <math.h>
struct point {double x; double y;};
void foo(struct point *p)
{ double d = sqrt((p->x)*(p->x) + (p->y)*(p->y));
  p->x /= d;
  p->y /= d;
}
int main()
{ struct point p1 = {2, 3};
  foo(&p1);
  printf("p1=(%5.2f, %5.2f)\n", p1.x, p1.y);
}
```

a) p1=(1.50, 1.75)      b) p1=(0.55, 0.83)  
c) p1=(2.00, 3.00)      d) p1=(2.00, 3.00)  
e) p1=(1.41, 1.73)

18

## Pointer and Index to the Same Place

```
int main()
{ char data[] = "Hello World";
  data[2] = 'X';
  char *linePt = &data[3];
  *linePt = 'Z';
  printf("[%s], [%s]\n", data, linePt);
}
```

output:

```
[HeXZo World], [Zo World]
```

19

## data[] as a Read Buffer

```
char data[MAX_CHARACTERS];
char* linePt = data;
int remainingBuffer = MAX_CHARACTERS;

while (fgets(linePt, remainingBuffer, inFile))
{ int lineLength = strlen(linePt);

  ...

  linePt += lineLength+1;
  remainingBuffer -= (lineLength+1)
}
```

20

## Quiz: Pointer and Index

```
int main()
{ char data[] = "Warcraft";
  data[7] = '+';
  char *linePt = &data[4];
  *linePt = '*';
  printf("[%s], [%s]\n", data, linePt);
}
```

- a) [Warcraft], [Warcraft]
- b) [Warcraf+], [Warc\*aft]
- c) [Warc\*aft], [Warcraf+]
- d) [Warc\*aft], [raf+]
- e) [Warc\*af+], [\*af+]