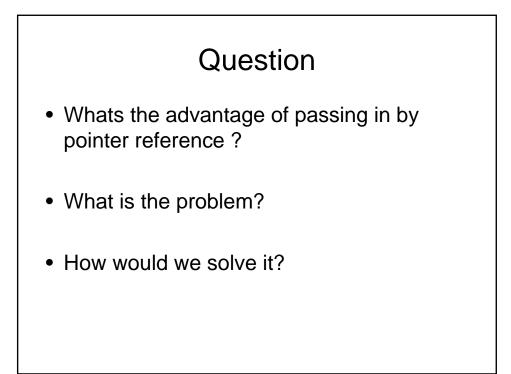
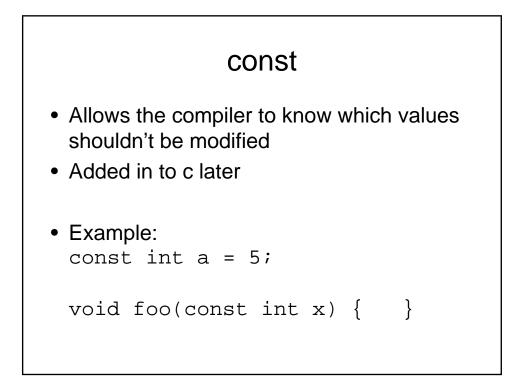
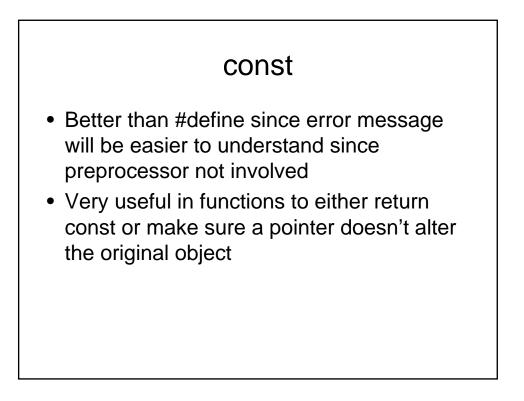
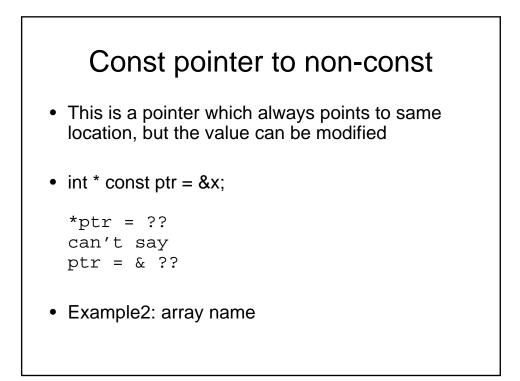


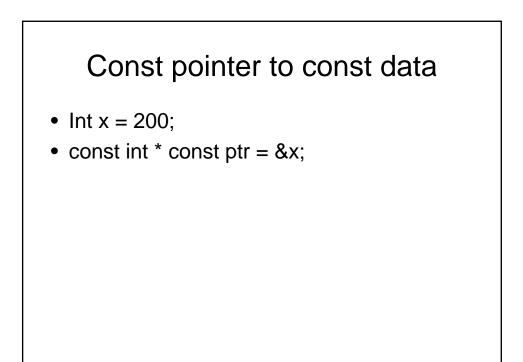
```
int main(){
    int number = 10;
    foo(&number);
    return 0;
}
void foo(int *p){
    *p = 30;
    }
```

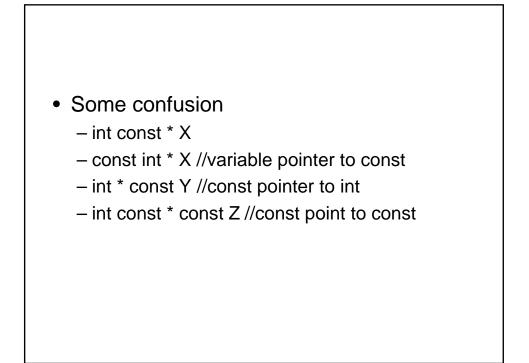


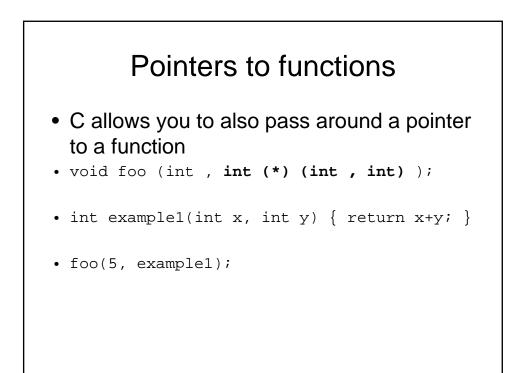




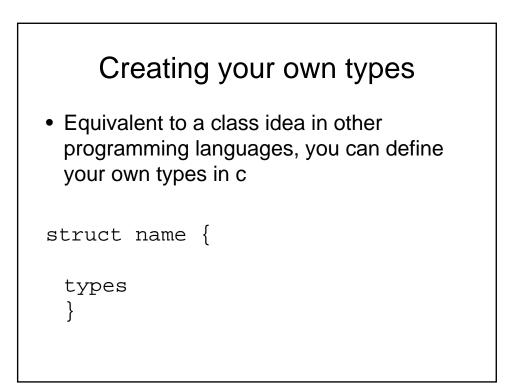






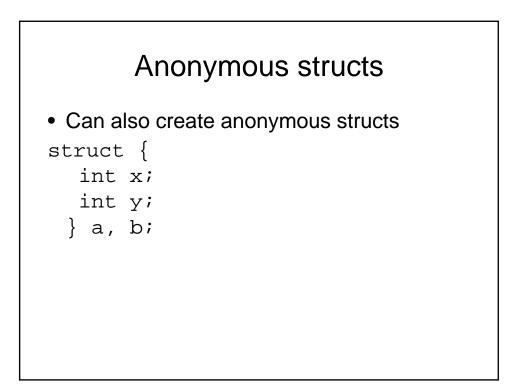


```
• void foo(int a, int (*A)(int,int)){
    if((*A)(5,10) > 0){
    }
    else {
    }
}
```



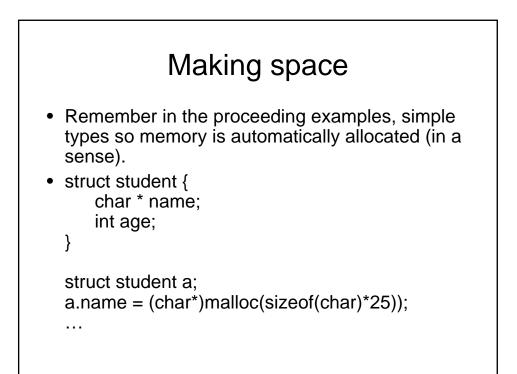
example

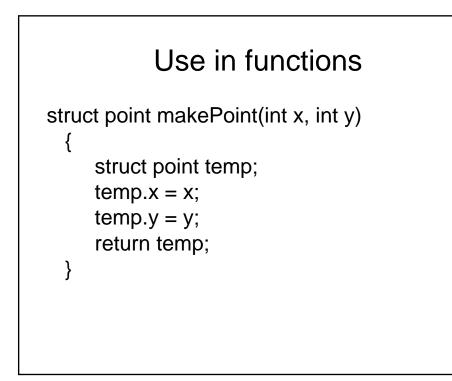
```
struct point {
    int x;
    int y;
    }
• Usage:
    struct point a;
    a.x = 5;
    a.y = 10;
```

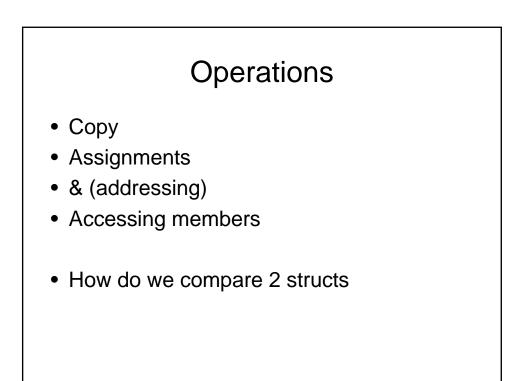


Nesting

```
struct rect {
    struct point pt1;
    struct point p2;
  }
• Use:
   struct rect largeScreen;
```







Structs and pointers

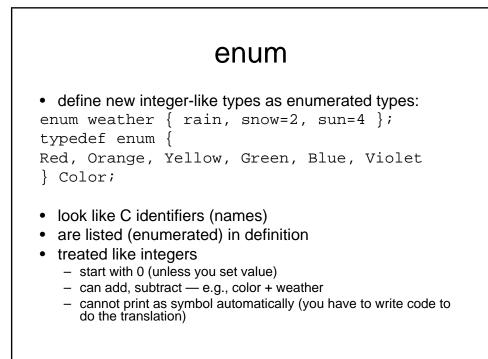
```
• struct point *example
= (struct point *)malloc(sizeof(struct
point));
```

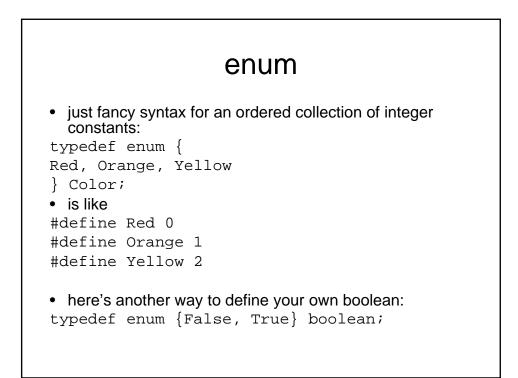
• (*example).x

what does
*example.x mean?

Shortcut: example->x

typedef short int smallNumber; typedef short int smallNumber; typedef unsigned char byte; typedef char String[100]; smallNumber x; byte b; String name;





Usage

```
enum Boolean {False, True};
```

. . .

. . .

```
enum Boolean shouldWait = True;
```

```
if(shouldWait == False) { .. }
```

```
struct
int main() {
struct {
    int x;
    char y;
    float z;
    } rec;
    rec.x = 3;
    rec.y = 'a';
    rec.z = 3.1415;
    printf( "rec = %d %c %f\n",rec.x,rec.y,rec.z
     );
    } // end of main()
```

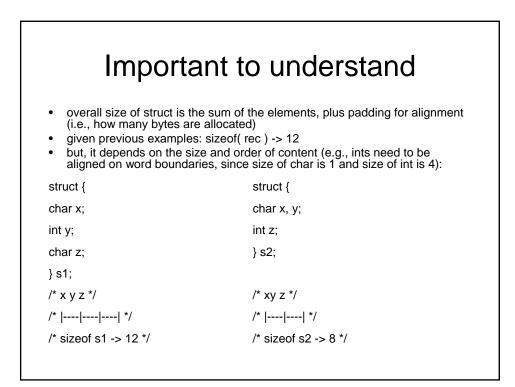
struct

```
int main() {
  struct record {
    int x;
    char y;
    float z;
    };
    struct record rec;
    rec.x = 3;
    rec.y = 'a';
    rec.z = 3.1415;
    printf( "rec = %d %c %f\n",rec.x,rec.y,rec.z );
    } // end of main()
```

```
int main() {
 typedef struct {
 int x;
 char y;
 float z;
 } RECORD;

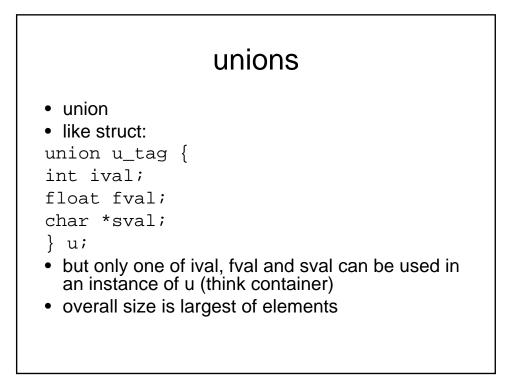
RECORD rec;
rec.x = 3;
rec.y = 'a';
rec.z = 3.1415;
printf( "rec = %d %c %f\n",rec.x,rec.y,rec.z );
} // end of main()
```

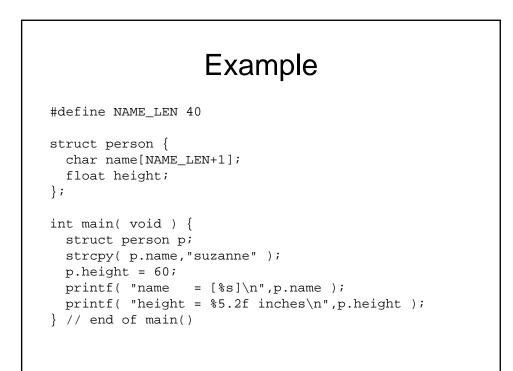
```
• note the use of malloc where "sizeof" takes the struct type as its
argument (not the pointer!)
int main() {
typedef struct {
int x;
char y;
float z;
} RECORD;
RECORD *rec = (RECORD *)malloc( sizeof( RECORD ));
rec->x = 3;
rec->y = 'a';
rec->z = 3.1415;
printf( "rec = %d %c %f\n",rec->x,rec->y,rec->z );
} // end of main()
```



Example in the second sec

Arrays of structs • notations for accessing elements: arr[i].field struct xyz { int x, y, z; }; struct xyz arr[2]; ... arr[0].x = 1; arr[0].y = 2; arr[0].z = 3; arr[1].x = 4; arr[1].y = 5; arr[1].z = 6;





For Next Class

- Do relevant reading
- Look over your exam, please see me if you don't understand/have questions
 - See you in lab Wednesday

Over view of assignment

- Extend the lab example
- Integrate perl in c and cgi
- Work with graphics
- Have something cool to show off to your friends or on interviews.
- Hints: if you are sending too much time....ask for help
 - examples