Week 3 Lecture 1

Revise Structures

Structures

Structures are related data

- A circle can be central point and a radius.
- The central point is described by two ints called x and y.
- The radius is describe by a float called radius.

```
struct circle {
  int x;
  int y;
  float radius;
};
```

- Whenever a struct circle is created a new element is created with all three values.

Access Structure Elements using Dot Notation

- Variable c
- X
- Y
- Radius

```
int main()
{
  struct circle c;

  c.x = 10;
  c.y = 20;
  c.radius = 1.5;

  printf("circle at (%d, %d) has radius %f\n",
         c.x, c.y, c.radius);
}
```

```
nat@wren:~/classes/sp/Week8/examples$ ./circle
circle at (10, 20) has radius 1.500000
```

Arrays of Structures

- You define an array of structures just like you define an array of anything else
  - E.g., `struct circle chain[80];`
  - This creates an array of 80 circles.
-

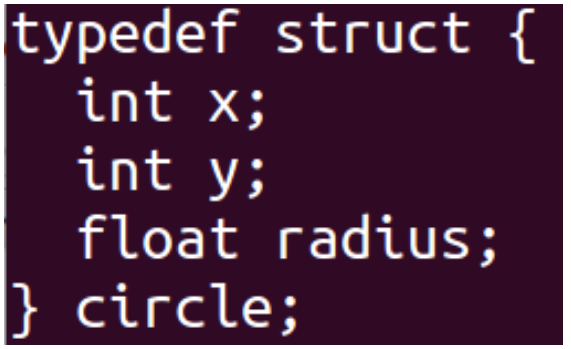
Set and Use Structure Arrays as usual

- Initialize the array
- Print the array

```
for (int i = 0; i < 10; i++) {
  chain[i].x = 10;
  chain[i].y = i;
  chain[i].radius = 1.5;
}
for (int i = 0; i < 10; i++) {
  printf("circle at (%d, %d) has radius %f\n",
         chain[i].x, chain[i].y, chain[i].radius);
}
```

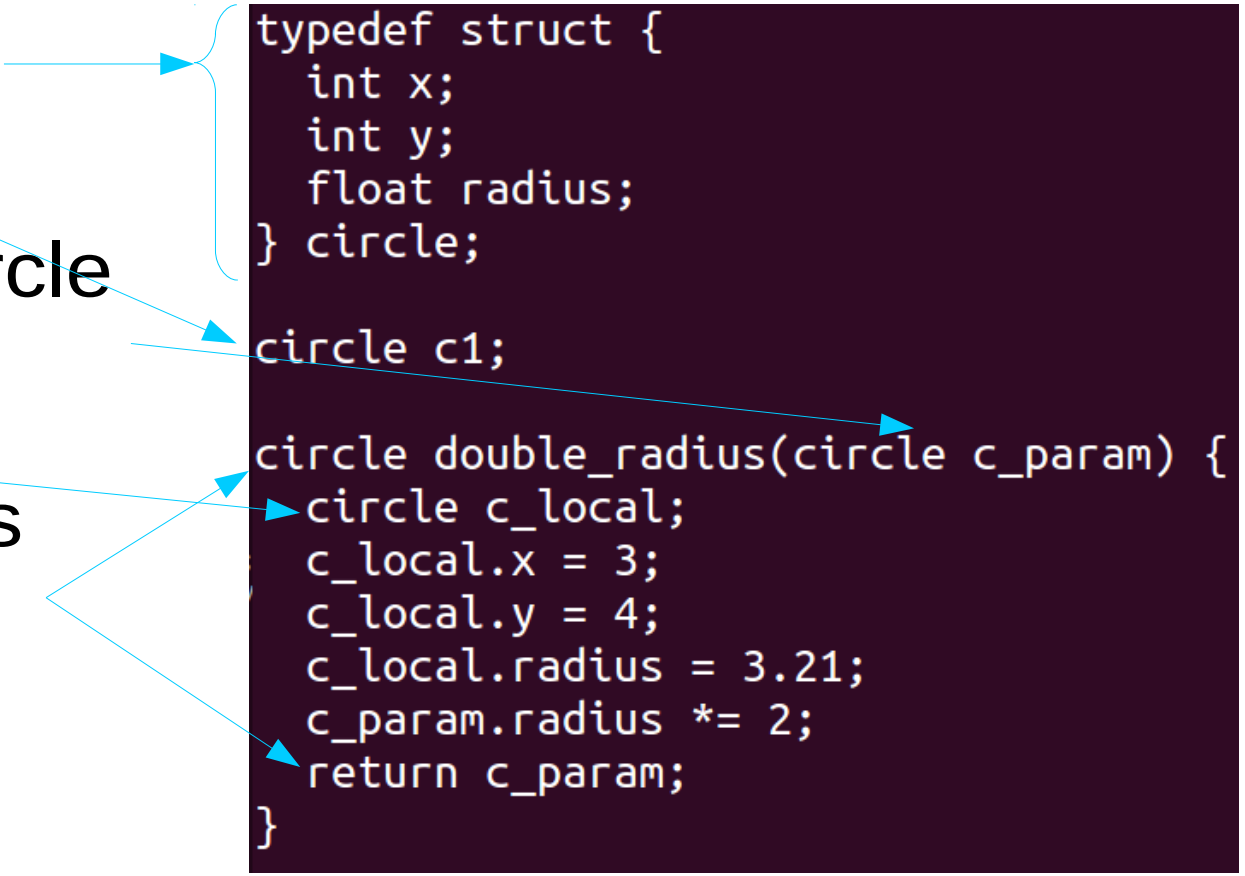
Define new types with typedef

- Type definition common with struct
- Define a circle type
  - Ints x and y
  - Float radius
- Allocates no memory



Define new defined type variable

- Type definition
- Global circle
- Parameter is circle
- Local circle
- Function returns circle



```
typedef struct {
  int x;
  int y;
  float radius;
} circle;

circle c1;

circle double_radius(circle c_param) {
  circle c_local;
  c_local.x = 3;
  c_local.y = 4;
  c_local.radius = 3.21;
  c_param.radius *= 2;
  return c_param;
}
```

Struct initialization

- Structs can be initialized.
- Easiest with defined type

```
typedef struct {
  int x;
  int y;
  float radius;
} circle;
```

```
main (int argc, char *argv[]) {
  circle c = {.x = 1, .y = 2, .radius = 1.23};
  printf ("circle at (%d, %d), %f\n", c.x, c.y, c.radius);
}
```

```
student@wren:~/sp/examples$ gcc -o struct struct.c
student@wren:~/sp/examples$ ./struct
circle at (1, 2), 1.230000
```

Structure contain Related Data

- E.g. Student
  - Serial number
  - Name
  - Grade

```
typedef struct student {
  int s_no;
  char *name;
  float grade;
} Student;
```

- Represents a real world object
  - A particular student
- Note: name is a string, that is, an array or characters, so structures can contain arrays.

Insert and Delete

Insertion and Deletion of Structs

- The insertion and deletion algorithms for structures are the same as for numbers
  - Structures can be assigned like numbers
  - The values from one are copied into the second

Algorithm Insertion (A[], N, val, loc)

Description-: A is an array having N elements. val is the value to be inserted at location loc.

1. if (loc > N)
   return   /*no room in array */

2. for j = N-1 to loc-1 /* count down to loc */
   2.1A[j+1] = A[j] /* move element up */

3. A[loc-1] = val /* put item in new loc */
4. N= N + 1 /* increase array size */

//Time Complexity-: O(N)

Algorithm Deletion (A[N], m, loc)

Description-: A is an array having N elements. loc is the location of the element to be deleted.

1. if (loc>N) then
   1.1 return
2. for j = loc to N-1 /* count up from loc */
   2.1 A[j-1] = A[j] /* move item down */
3. N = N – 1; /* reduce array size */

//Time Complexity-: O(N)