Lecture 11:
Structs, stack, and dynamic memory

CSE 29: Systems Programming and Software Tools

Aaron Schulman (Shalev)

UCSDCSE
Computer Science and Engineering

Introducing struct datatypes

- Up until now all variables have been ether single elements or arrays

```
int latitude;
int longitude;
```

- struct is a datatype that can combine elements into one variable

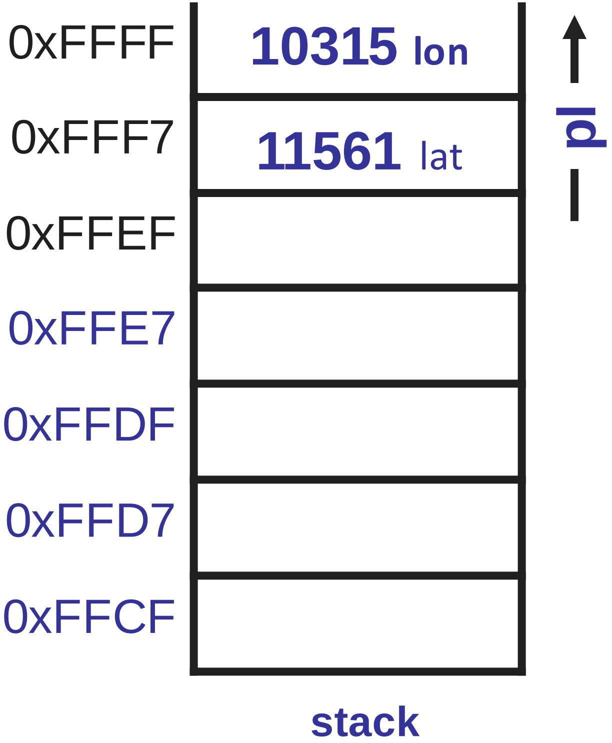
```
struct place {
     int latitude;
     int longitude;
};
```

Data members (or member variables)

Using struct datatypes

```
 struct place {
        long int lat; // Latitude
        long int lon; // Longitude
 };

int main() {
 struct place pl;
 pl.lat = 10315;
 pl.lon = 11561;
 return 0;
}
```



0xFFFF	10315 lon
0xFFF7	11561 lat
0xFFEF	
0xFFE7	
0xFFDF	
0xFFD7	
0xFFCF	

Passing a struct to a function

```
long int distance(struct place p1, struct place p2) {
    // Compute the distance from p1 to p2
    return ((p1.lat – p2.lat)**2); // not actual distance
}

int main() {
  struct place dca = {389072, -770369};
  struct place san = {32.7157, -1171611};
  long int dist = distance(dca, san); // structs will be copied ☹
}                                 // now two copies on stack
```

Dynamic memory allocation

We have been statically initializing the size of an array at *compile time*:

```
int arr[4]; // The length 4 is defined at compile time
```

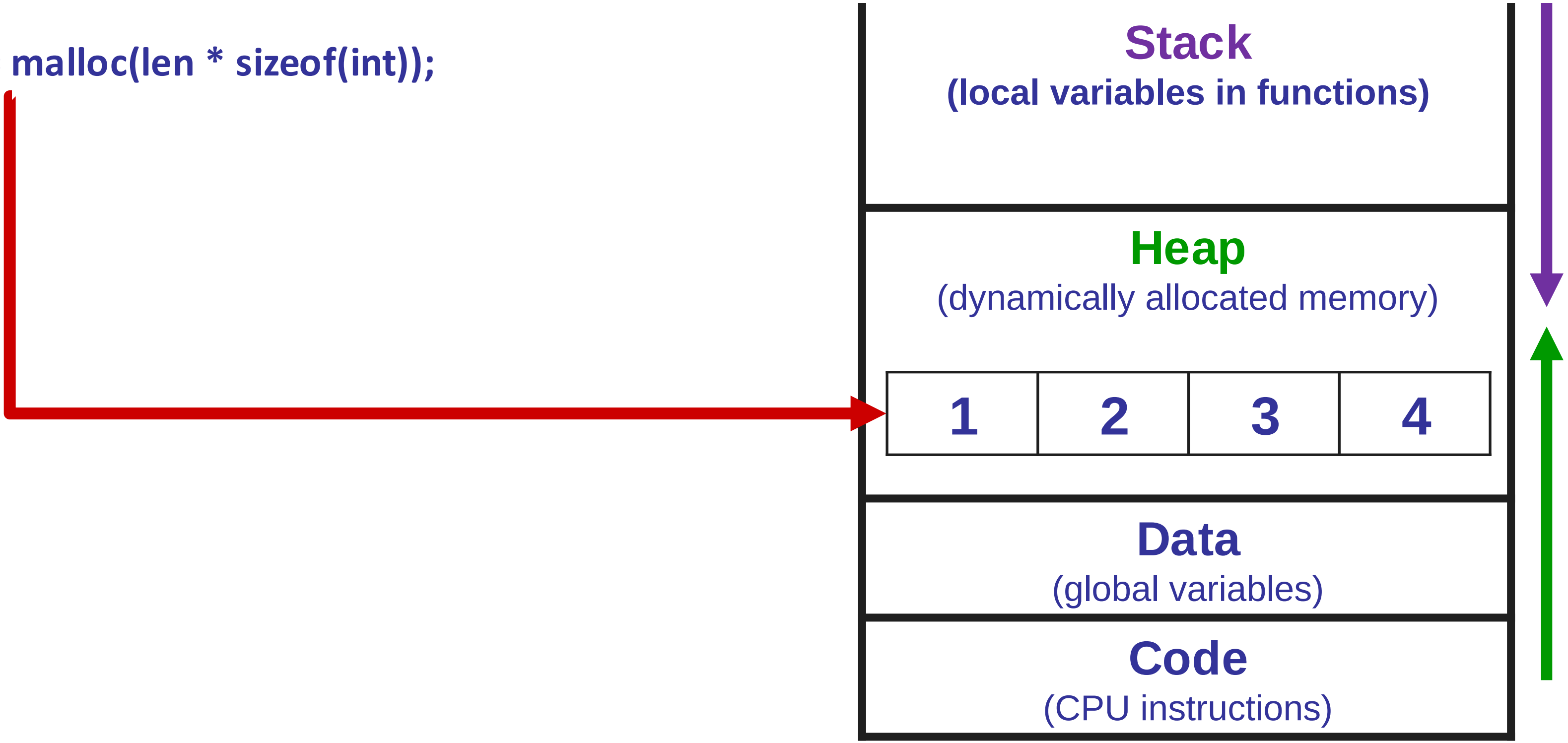
What if we want to create an array where the size is defined at *runtime*?

```
int len = 10; // Can be changed at runtime
int *iarr = NULL;
iarr = malloc(len * sizeof(int));

iarr[8] = 5;  // or *(iarr + 8) = 5
```

Where are malloc() allocations in memory?

```
int iarr* = malloc(len * sizeof(int));
```


Stack
(local variables in functions)
Heap
(dynamically allocated memory)
1
2
3
4
Data
(global variables)
Code
(CPU instructions)

Dynamic memory allocation

- Do you think malloc() will allocate memory for arrays from the stack?
  - Why or why not?

- When you compile a function, you know what the size of all variables are:
  - The compiler automatically makes room for them on the stack
    - This is why there is no pointer to an array stored on the stack!

- When you don’t know the size of variables at compile time, you need to allocate them from another memory region
  - The Heap: Memory region in a program for dynamically sized variables/arrays
    - The heap will need to be managed (later in this class!)