

Lecture 10:

Dynamic Memory Allocation (malloc/heap)

CSE 29: Systems Programming and Software Tools
Aaron Schulman (Shalev)



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 = 40; // Can be set at runtime
```

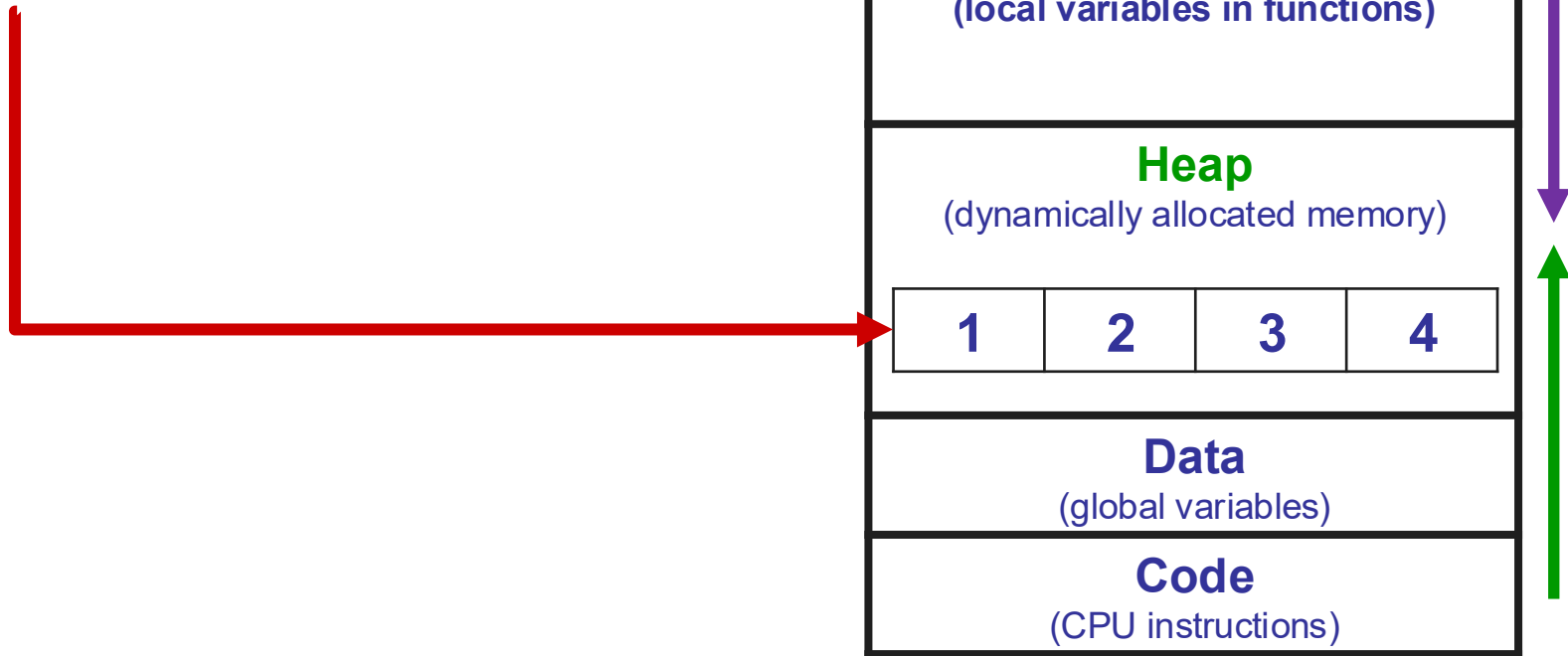
```
char* darr = NULL;
```

```
darr = malloc(len * sizeof(char));
```

```
darr[8] = 'c'; // or *(darr + 8) = 'c'
```

Where are malloc() allocations in memory?

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



Dynamic memory allocation

- ❑ You do not always need to use malloc() when you need to store arrays!
- ❑ When you know what the size of array is, use the stack:
 - ❑ The compiler automatically makes room for them on the stack
- ❑ When you don't know the size of arrays at compile time, you need to allocate them from another memory region
 - ❑ **The Heap:** Memory region in a program for dynamically sized arrays
 - » The heap will need to be managed (you will implement this in this class!)
 - » You can change the amount of memory used to store an array during the execution of your program

When is allocated memory available again to reuse for other allocations?

Stack (*Static* allocation): As soon as the function ends the variables are no longer considered used on the stack

- ▣ You do not need to do anything to free a variable on the stack

Heap (*Dynamic* allocation): The programmer must *manually* **free(ptr)** the allocation

- ▣ **How can you guarantee that there is no other code that will use a pointer again?**
 - » Pointers can be copied as variables in functions, returned from functions, put into arrays.
- ▣ One of the hardest problems in computer science is automatically determining when it's ok to **free()**
 - » Java/Python's garbage collection, Rust's ownership model
- ▣ If a pointer is no longer accessible, you may **never** be able to free it during the program!
 - » This is called the dreaded: "Memory Leak"



Review Questions

1. 1. When do you need dynamic memory allocation vs. a static array?
2. 2. Write a line that will `malloc()` an array that can hold 10 `uint32_t` numbers.
3. 3. When should you `free()` a variable that has been dynamically allocated?