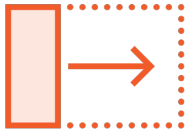
Managing Data and Memory Allocation in C

DYNAMICALLY ALLOCATING MEMORY USING C

Zachary Bennett
SOFTWARE ENGINEER

@z_bennett_ github.com/zbennett10

What Is Computer Memory (RAM)

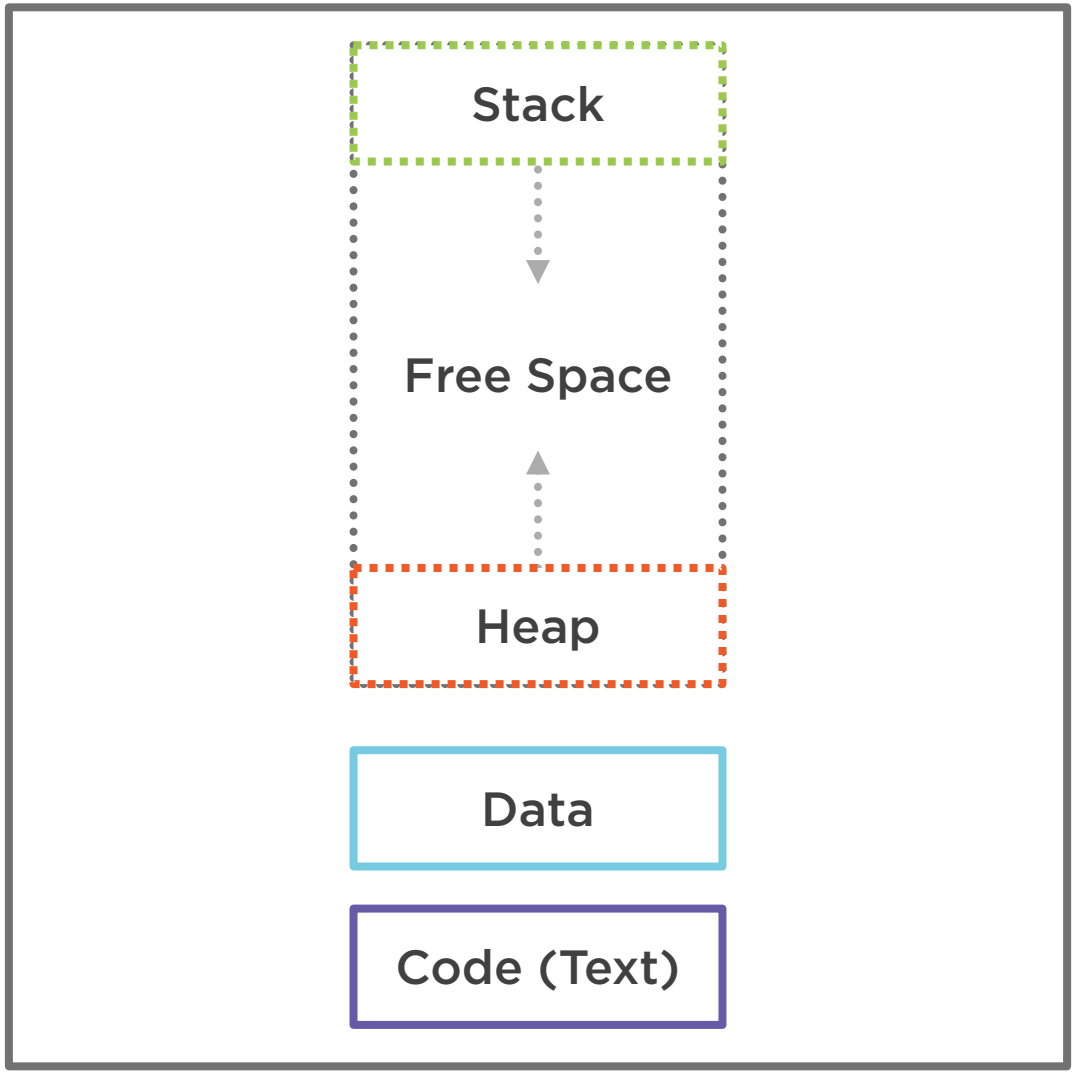


Speed is the name of the game

The CPU spends less time transferring data

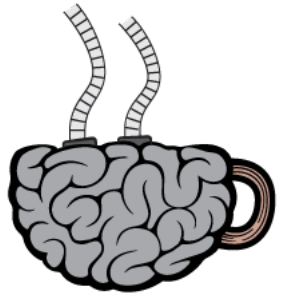
Core to working with and understanding C

Memory Segments



Dynamic vs. Static Memory Allocation

Static Memory Allocation	Dynamic Memory Allocation
Allocated on the stack	Allocated on the heap
Memory allocated at compile time	Memory allocated at runtime
Memory cannot be freed	Memory can be freed



WIRED BRAIN
COFFEE

Embedded C Programs

Wired Brain Coffee is having problems with memory management in their embedded C programs.

Fixed Memory

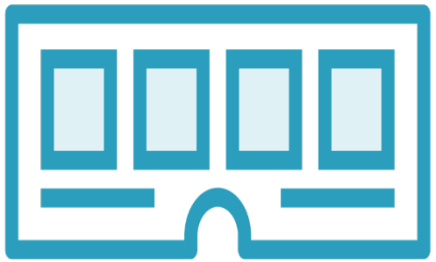
There is not very much memory available to C programs.

Dynamic Environment

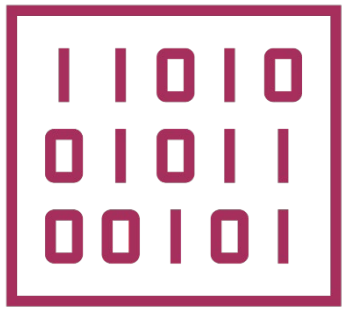
The amount of memory needed for the program to run cannot be determined at compile time.

C Standard Library Core Memory Functions

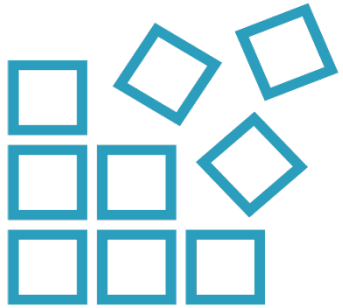
<stdlib.h>



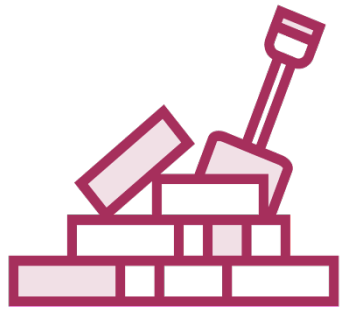
malloc



calloc



realloc



free

malloc

```
void *malloc( size_t byte_size )
```

```
int *pointer  = (int*)malloc( sizeof( int ) );
```

Function Signature

- Takes an unsigned, 32-bit integer as its sole argument
- Returns a pointer to void in anticipation of a cast

Example usage

- It is a best practice to use “sizeof” in order to determine the correct byte size of the type that is passed into it.

calloc

```
void *calloc( size_t count,  size_t byte_size )
```

```
int *pointer  = (int*)calloc( 1, sizeof( int ) );
```

Function Signature

- Takes two, unsigned 32-bit integers as arguments
- Allocates and then initializes memory block to 0
- Like malloc, returns a pointer to void in anticipation of a cast

Example usage

- Unlike malloc, calloc has two explicit arguments
- “calloc” is a slower than malloc

realloc

```
void *realloc( void *ptr, size_t byte_size )
```

```
ptr  = (int*)realloc( ptr, sizeof( int ) * 2 );
```

Function Signature

- Takes a generic pointer and an unsigned, 32-bit integer as an argument
- Returns a pointer to void in anticipation of a cast

Example usage

- You can use “realloc” to resize the amount of memory allocated on the heap to a given pointer.

free

```
void free( void *ptr )
```

```
int *ptr  = (int*)malloc( sizeof( int ) );

free(ptr);
```

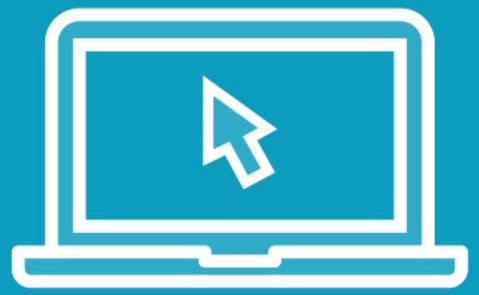
Function Signature

- Takes a pointer of any type as its sole argument

Example usage

- Using “free” will free up previously allocated memory on the heap.

Demo



Including the <stdlib.h> header file

How to allocate and resize memory

- Allocate memory on the heap with “malloc”
- Allocate and initialize memory with “calloc”
- Resize memory with “realloc”

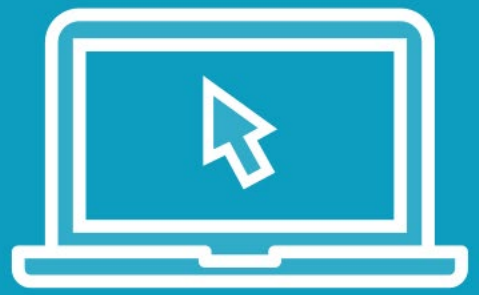
Freeing up memory on the heap

“With great power comes great responsibility.”

Uncle Ben from “Spider-Man”

Dynamic Memory Allocation Is Dangerous

Demo

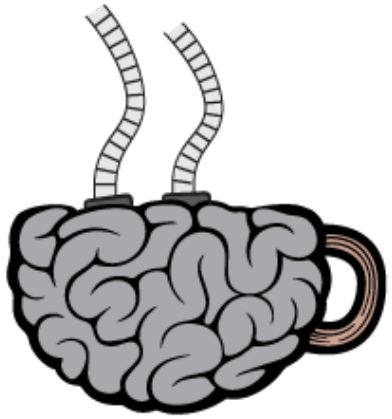


Crucial to see mishaps from the get-go

Examples of dynamic memory dangers

- Dangling pointers
- Double frees
- Forgetting to use “sizeof”
- NULL pointers returned from malloc/calloc/realloc

Call To Action



WIRED BRAIN

COFFEE

We need your help!

Overview/ Summary

Four primary memory segments

Static and dynamic memory allocation

Dynamic memory allocation using the <stdlib.h> memory functions

Avoid memory leaks!