Dynamic memory allocation

Adapted from materials by Dr. Carrier

Mandy Jouan - Flickr

So you want to allocate memory…

So you want to allocate memory…

`malloc(size)`

- Allocates `size` bytes
- On the heap
- Contiguous

So you want to allocate memory…

```
malloc(size)
```

- Allocates `size` bytes
- On the heap
- Contiguous

```
void* malloc(size);
```

- Returns a void pointer
    - Think of it like a generic pointer
    - Needs to be cast to the type you need
- Pointer points to start of allocated block

So you want to allocate memory…

```
malloc(size)
```

- Allocates `size` bytes
- On the heap
- Contiguous

```
void* malloc(size);
```

- Returns a void pointer
  - Think of it like a generic pointer
  - Needs to be cast to the type you need
- Pointer points to start of allocated block

Example:

```
int* p = (int*)malloc(4 * 100);
```

Includes

Note, malloc (and those like it), require a new include:

```
#include <stdlib.h>
```

Using specific types

What if we don’t know the size of our type?

Using specific types

What if we don’t know the size of our type?

Use sizeof!

```
long* p = (long*)malloc(sizeof(long) * 10);
```

Alternatives…

`calloc(count, size);`

- Allocates count * size bytes
- Initializes values to zero (malloc does not)

```
float* p = calloc(64, sizeof(float));
```

What if we want MORE space?

Imagine we have an array of 16 ints.

Now we need to double it to 32.

But we don’t want to copy them by hand…

What if we want MORE space?

Imagine we have an array of 16 ints.

Now we need to double it to 32.

But we don’t want to copy them by hand…

```
int* p = (int*)malloc(sizeof(int) * 16);
```

```
p = realloc(p, sizeof(int) * 32);
```

What if we want MORE space?

Imagine we have an array of 16 ints.

Now we need to double it to 32.

But we don’t want to copy them by hand…

```
int* p = (int*)malloc(sizeof(int) * 16);
```

```
p = realloc(p, sizeof(int) * 32);
```

- Allocates new memory
- Copies the old over
- “Frees” old memory
- Not guaranteed to initialize new mem to zero

What about our old memory?

We need to *free* it when we’re done!

```
free(pointer);
```

If we forget, we can have memory leaks!