Dynamic Memory Allocation

Dynamic Memory Allocation

- We've seen one way to work with pointers, namely pointing a pointer variable at another variable that already exists in our system.
  - This requires us to know exactly how much memory our system will need at the moment our program is compiled.

- What if we **don't** know how much memory we'll need at compile-time? How do we get access to new memory while our program is running?

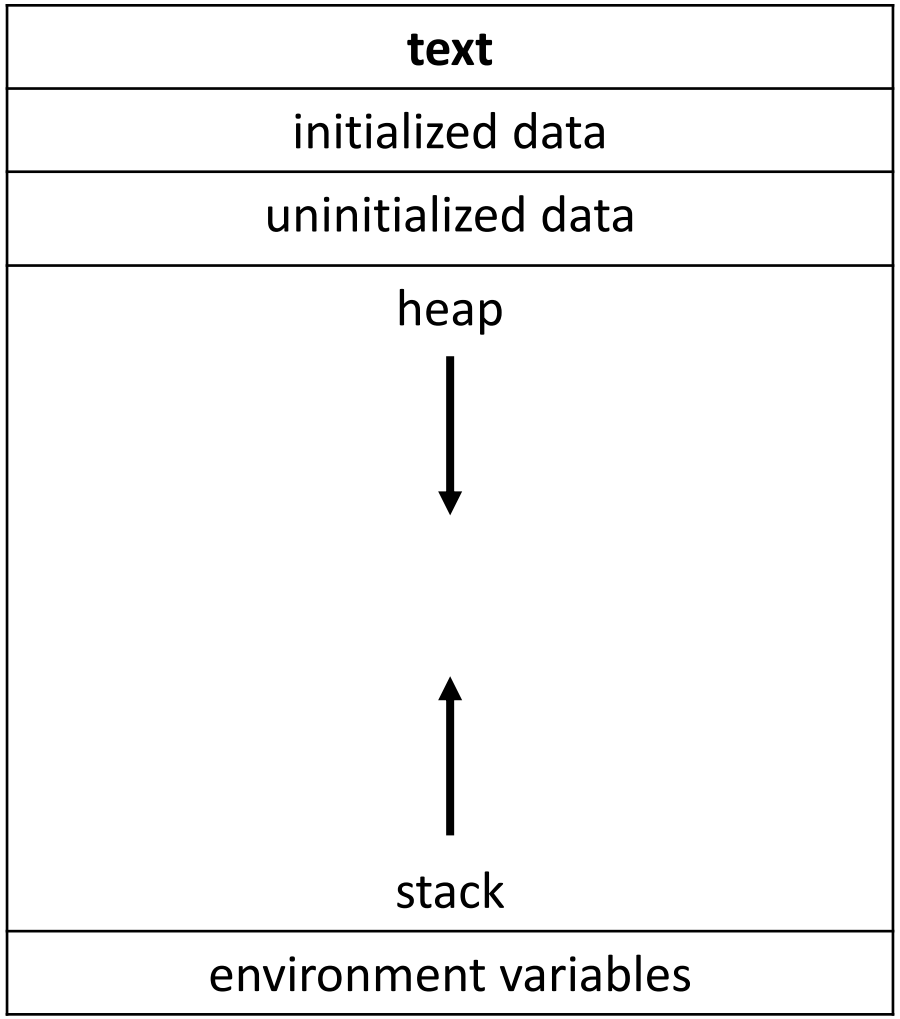
Dynamic Memory Allocation

- We can use pointers to get access to a block of **dynamically-allocated memory** at runtime.

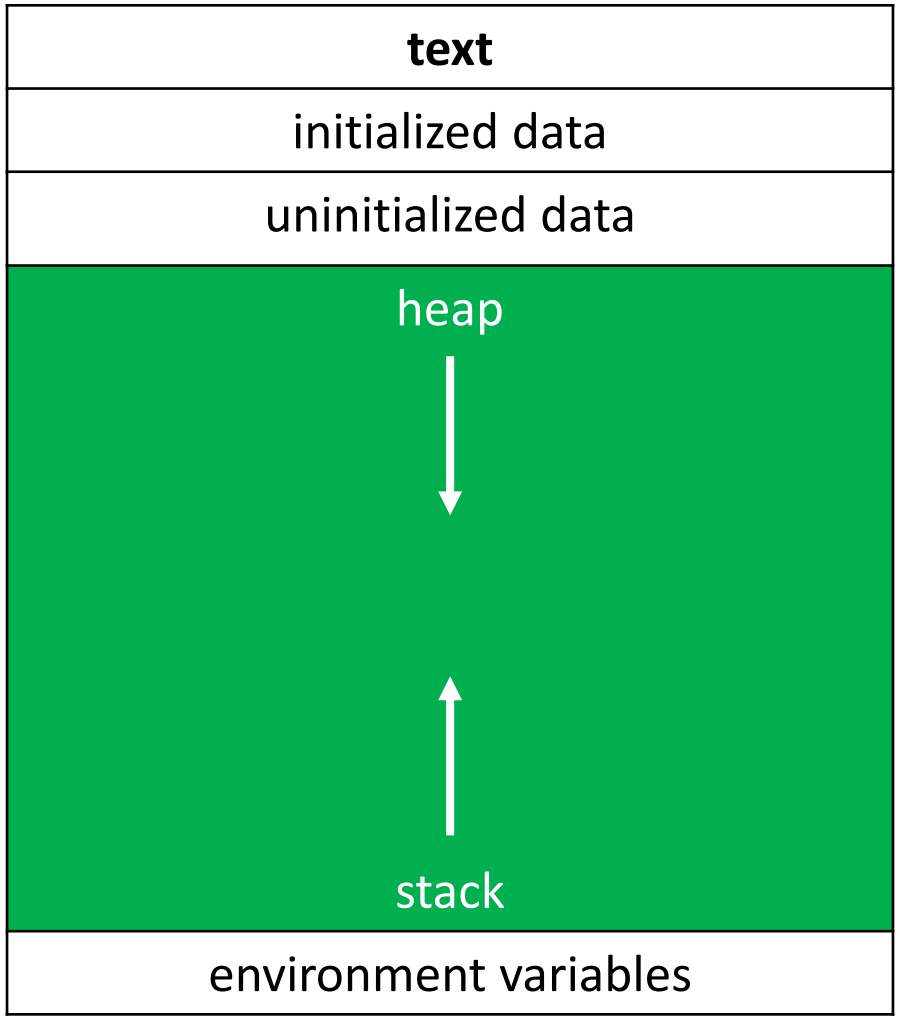
- Dynamically allocated memory comes from a pool of memory known as the **heap**.

- Prior to this point, all memory we've been working with has been coming from a pool of memory known as the **stack**.

Dynamic Memory Allocation



Dynamic Memory Allocation



Dynamic Memory Allocation

- We get this dynamically-allocated memory by making a call to the C standard library function `malloc()`, passing as its parameter the number of bytes requested.

- After obtaining memory for you (if it can), `malloc()` will return a pointer to that memory.

- What if `malloc()` **can't** give you memory? It'll hand you back `NULL`.

Dynamic Memory Allocation

```
// statically obtain an integer
int x;
```

Dynamic Memory Allocation

```
// statically obtain an integer
int x;

// dynamically obtain an integer
int *px = malloc(4);
```

Dynamic Memory Allocation

```
// statically obtain an integer
int x;

// dynamically obtain an integer
int *px = malloc(sizeof(int));
```

Dynamic Memory Allocation

```
// get an integer from the user
int x = GetInt();
```

Dynamic Memory Allocation

```
// get an integer from the user
int x = GetInt();

// array of floats on the stack
float stack_array[x];
```

Dynamic Memory Allocation

```
// get an integer from the user
int x = GetInt();

// array of floats on the stack
float stack_array[x];

// array of floats on the heap
float* heap_array = malloc(x * sizeof(float));
```

Dynamic Memory Allocation

- Here's the trouble: Dynamically-allocated memory is not automatically returned to the system for later use when the function in which it's created finishes execution.

- Failing to return memory back to the system when you're finished with it results in a **memory leak** which can compromise your system's performance.

- When you finish working with dynamically-allocated memory, you must `free()` it.

Dynamic Memory Allocation

```
char* word = malloc(50 * sizeof(char));
```

Dynamic Memory Allocation

```
char* word = malloc(50 * sizeof(char));

// do stuff with word
```

Dynamic Memory Allocation

```
char* word = malloc(50 * sizeof(char));

// do stuff with word

// now we're done working with that block
free(word);
```

Dynamic Memory Allocation

- Three golden rules:

1. Every block of memory that you `malloc()` must subsequently be `free()`d.

2. Only memory that you `malloc()` should be `free()`d.

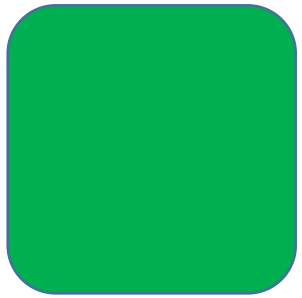
3. Do not `free()` a block of memory more than once.

Dynamic Memory Allocation

```
int m;
```

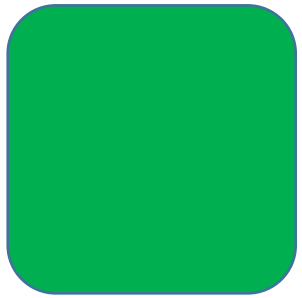
Dynamic Memory Allocation

```
int m;
```



Dynamic Memory Allocation

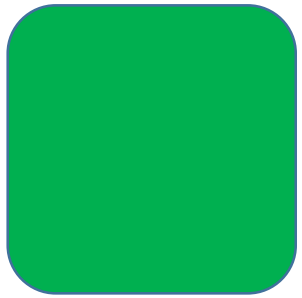
```
int m;
int* a;
```



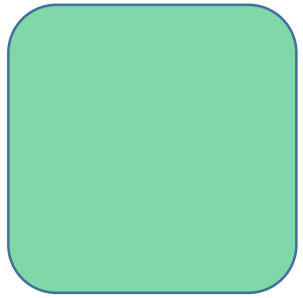
m

Dynamic Memory Allocation

```
int m;
int* a;
```



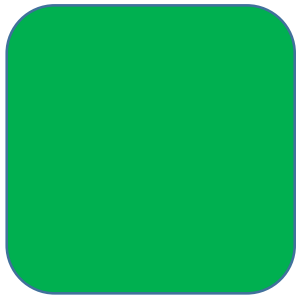
m



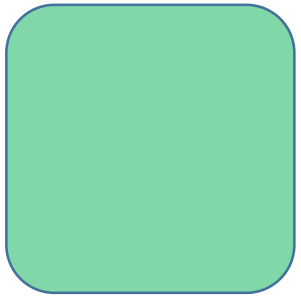
a

Dynamic Memory Allocation

```
int m;
int* a;
int* b = malloc(sizeof(int));
```



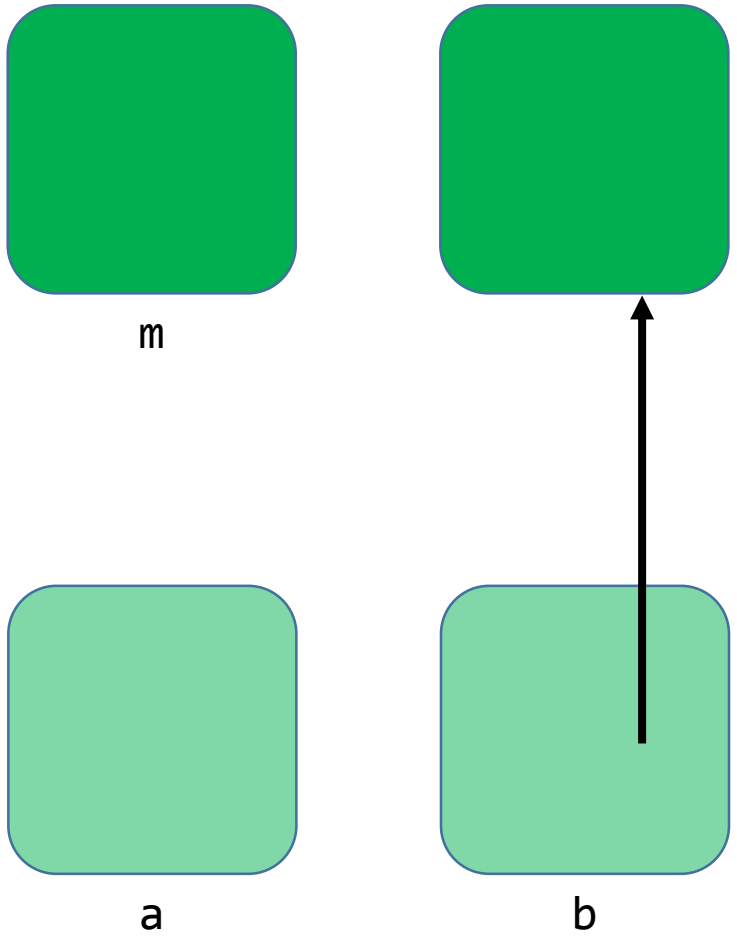
m



a

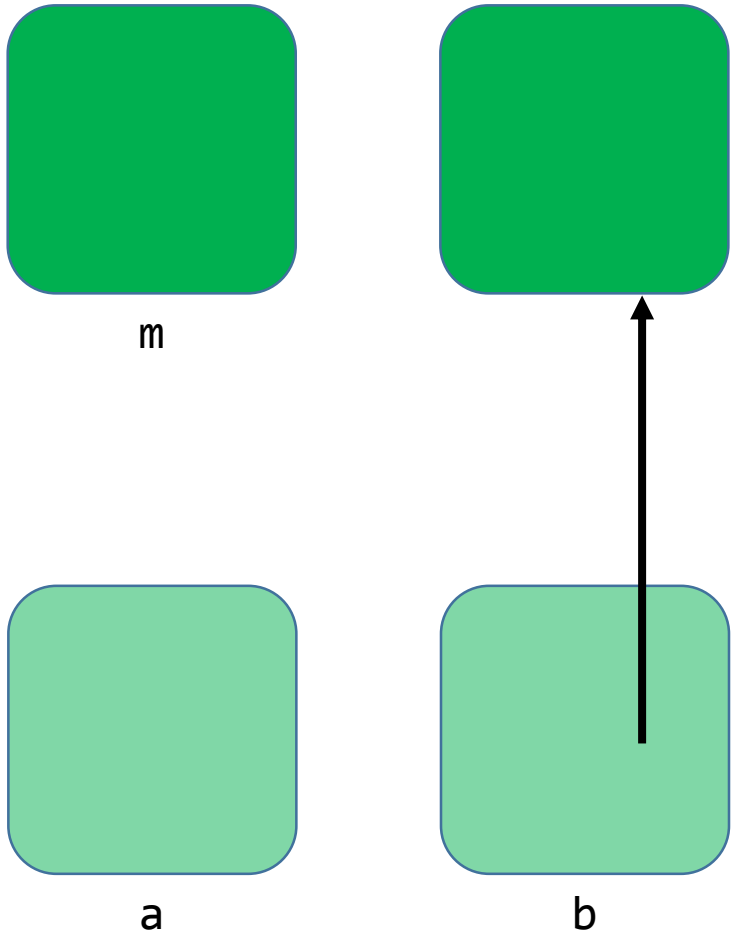
Dynamic Memory Allocation

```
int m;
int* a;
int* b = malloc(sizeof(int));
```



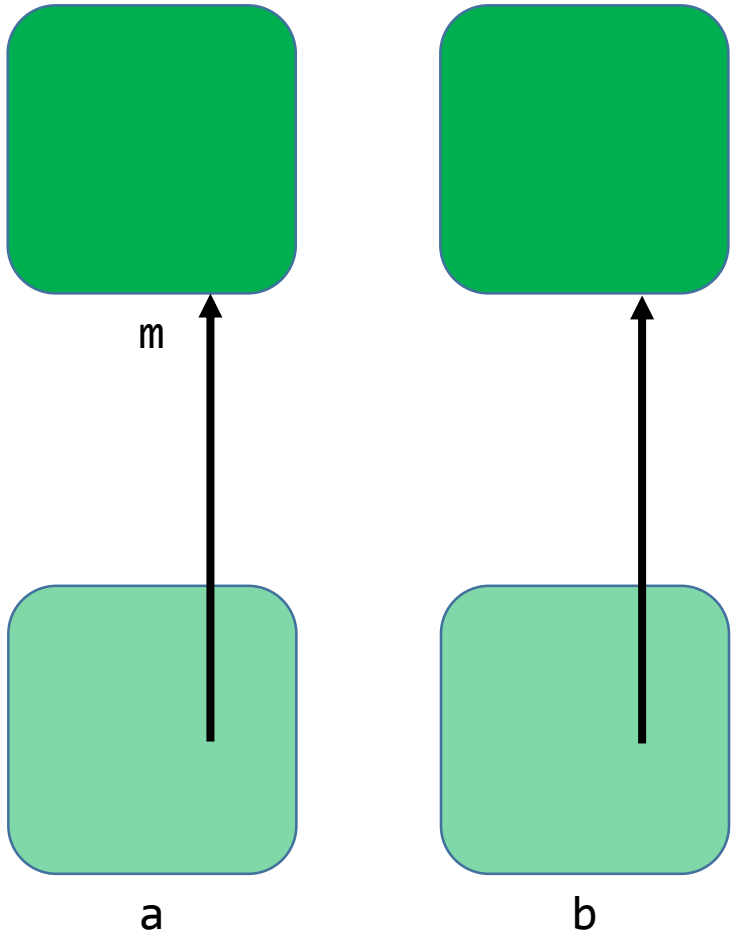
Dynamic Memory Allocation

```
int m;
int* a;
int* b = malloc(sizeof(int));
a = &m;
```



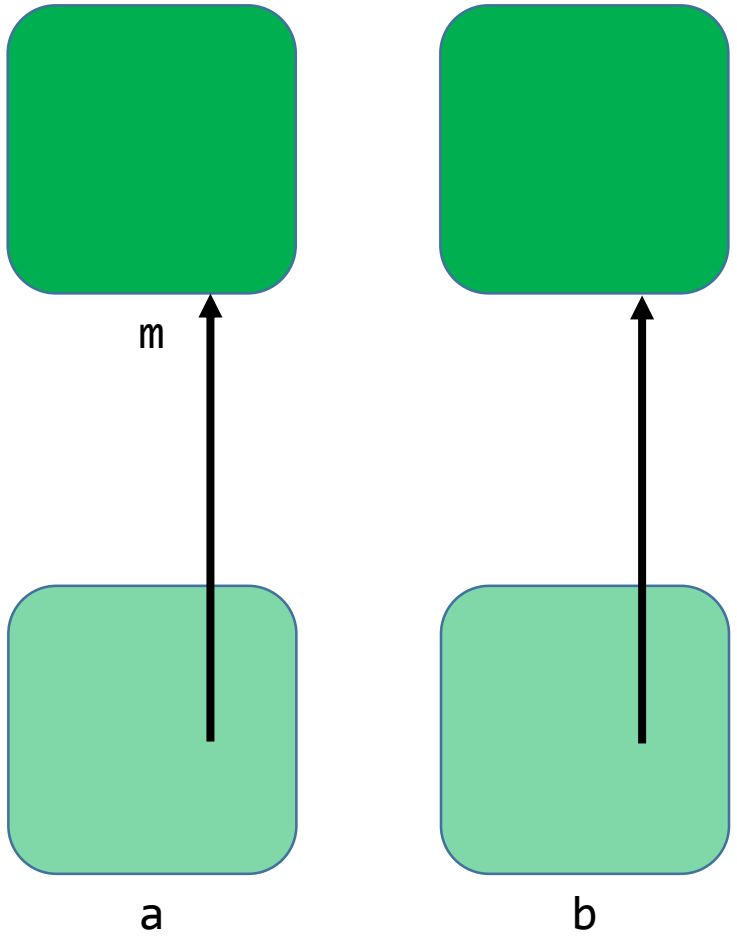
Dynamic Memory Allocation

```
int m;
int* a;
int* b = malloc(sizeof(int));
a = &m;
```



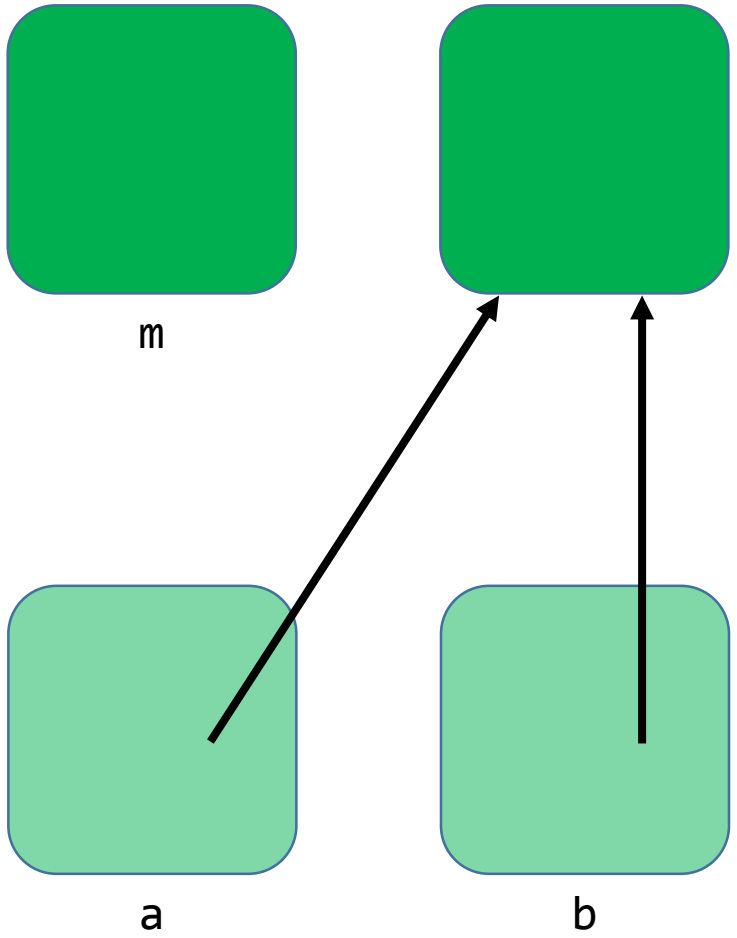
Dynamic Memory Allocation

```
int m;
int* a;
int* b = malloc(sizeof(int));
a = &m;
a = b;
```



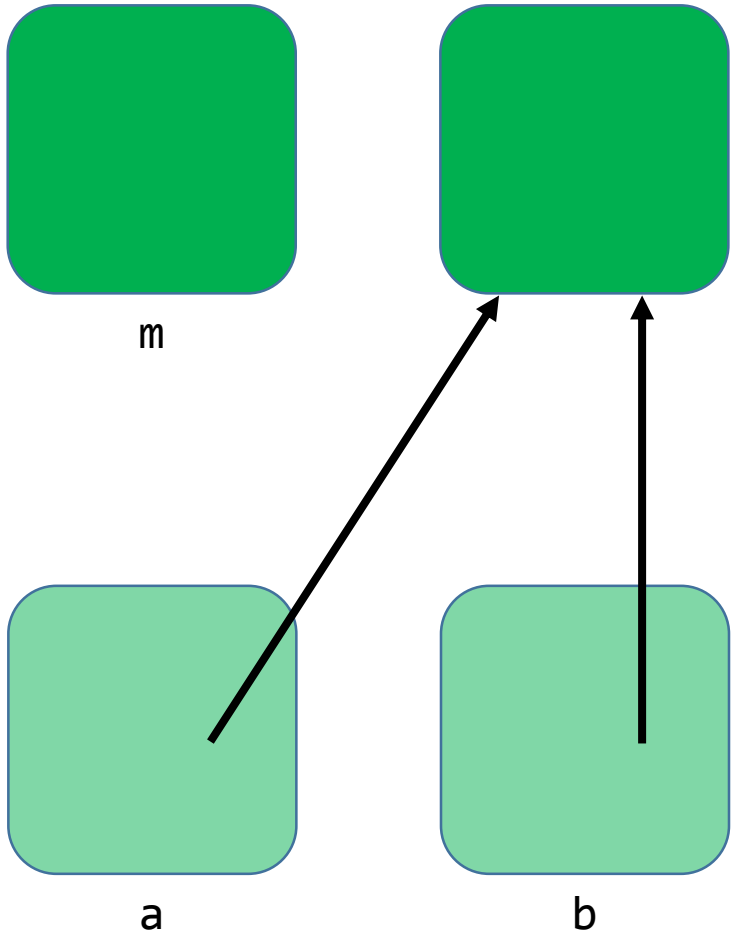
Dynamic Memory Allocation

```
int m;
int* a;
int* b = malloc(sizeof(int));
a = &m;
a = b;
```



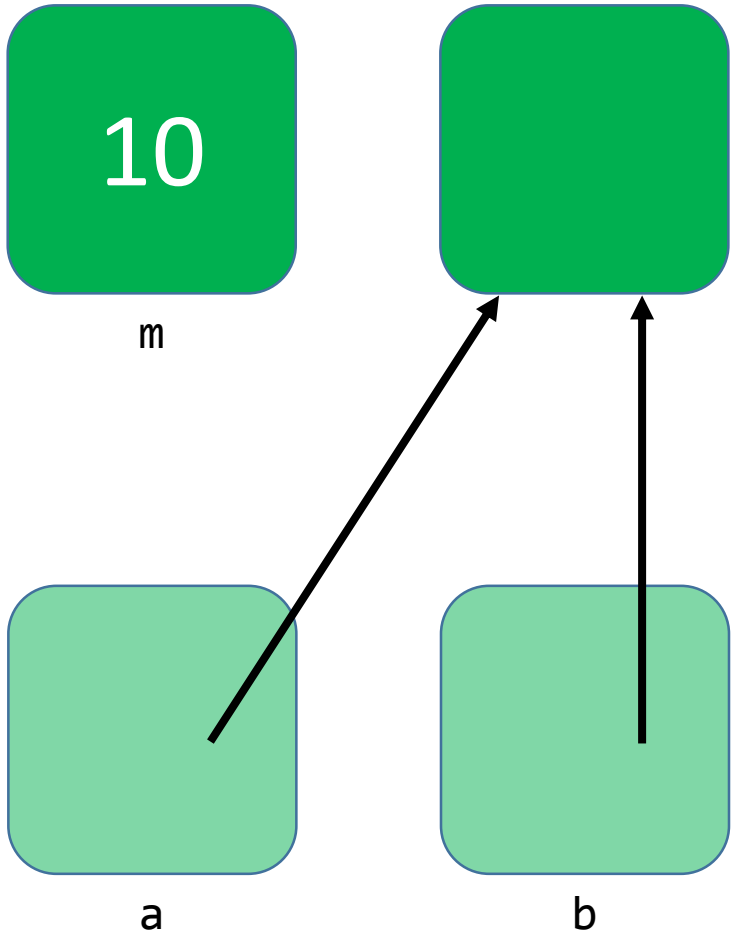
Dynamic Memory Allocation

```
int m;
int* a;
int* b = malloc(sizeof(int));
a = &m;
a = b;
m = 10;
```



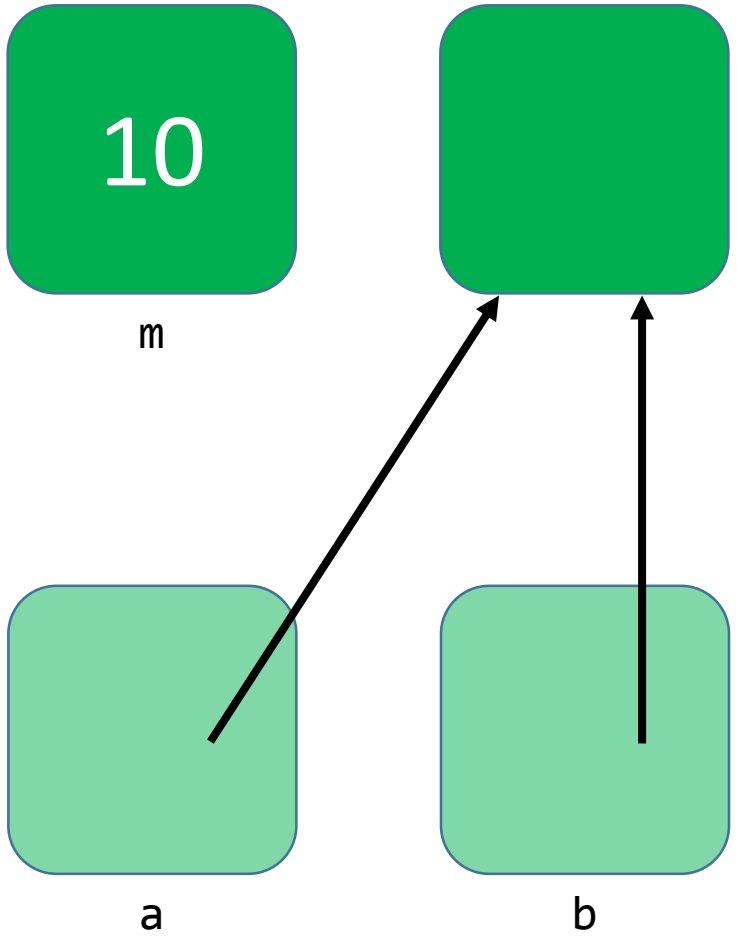
Dynamic Memory Allocation

```
int m;
int* a;
int* b = malloc(sizeof(int));
a = &m;
a = b;
m = 10;
```



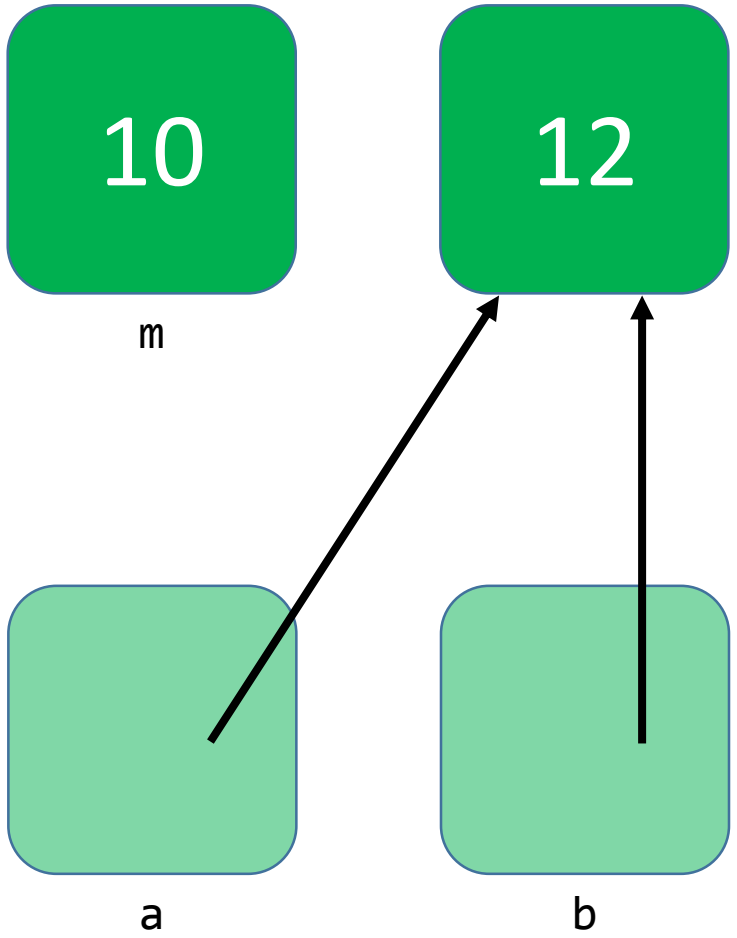
Dynamic Memory Allocation

```
int m;
int* a;
int* b = malloc(sizeof(int));
a = &m;
a = b;
m = 10;
*b = m + 2;
```



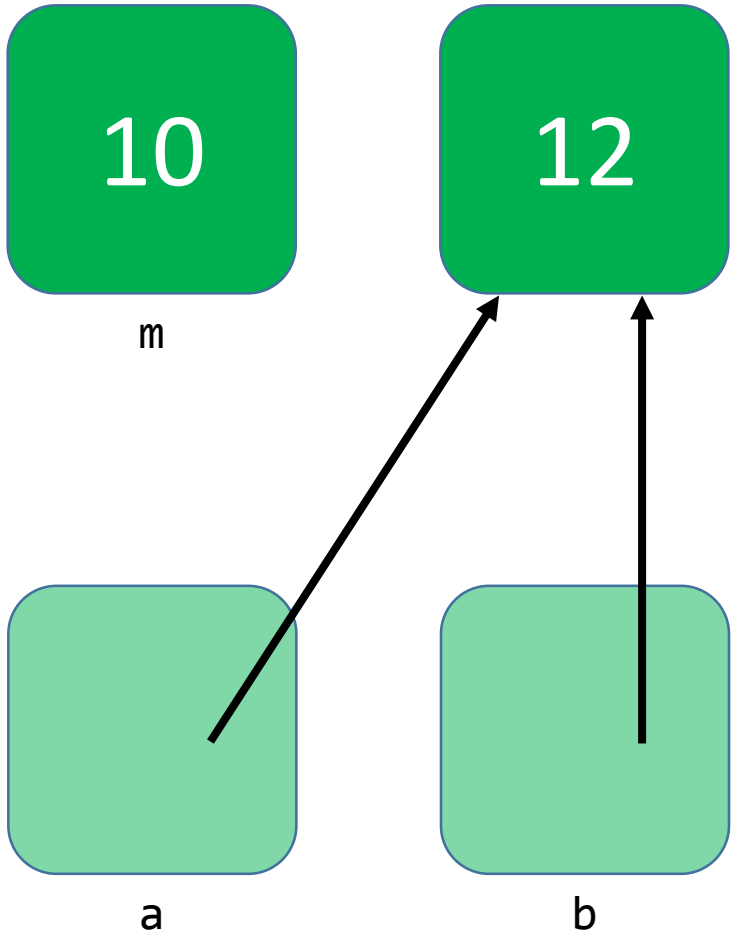
Dynamic Memory Allocation

```
int m;
int* a;
int* b = malloc(sizeof(int));
a = &m;
a = b;
m = 10;
*b = m + 2;
```



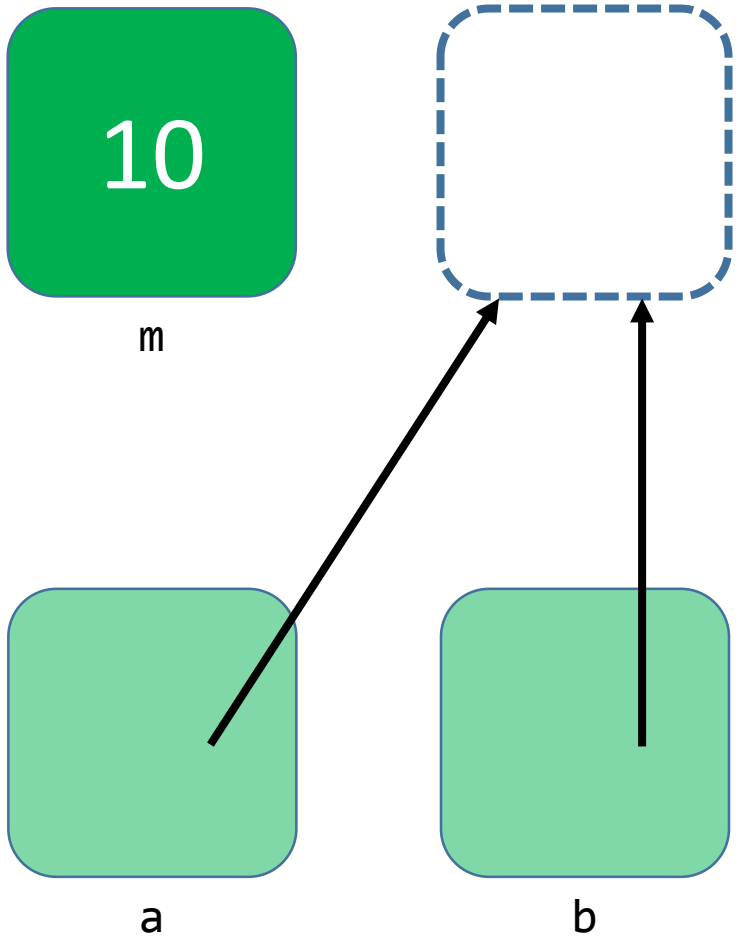
Dynamic Memory Allocation

```
int m;
int* a;
int* b = malloc(sizeof(int));
a = &m;
a = b;
m = 10;
*b = m + 2;
free(b);
```



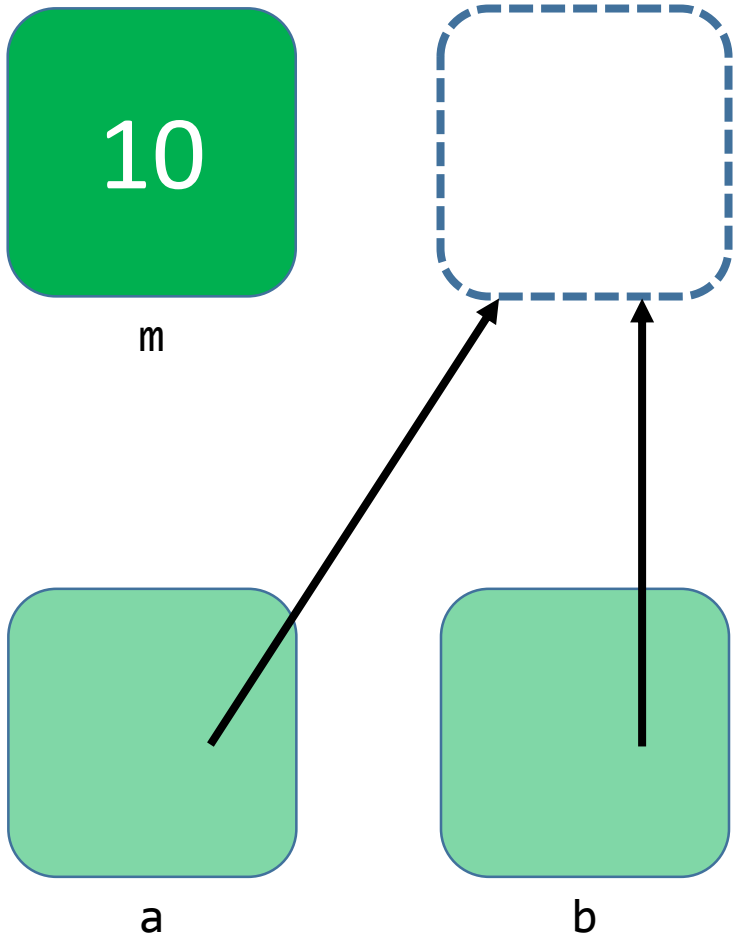
Dynamic Memory Allocation

```
int m;
int* a;
int* b = malloc(sizeof(int));
a = &m;
a = b;
m = 10;
*b = m + 2;
free(b);
```



Dynamic Memory Allocation

```
int m;
int* a;
int* b = malloc(sizeof(int));
a = &m;
a = b;
m = 10;
*b = m + 2;
free(b);
*a = 11;
```



Dynamic Memory Allocation

```
int m;
int* a;
int* b = malloc(sizeof(int));
a = &m;
a = b;
m = 10;
*b = m + 2;
free(b);
*a = 11;
```