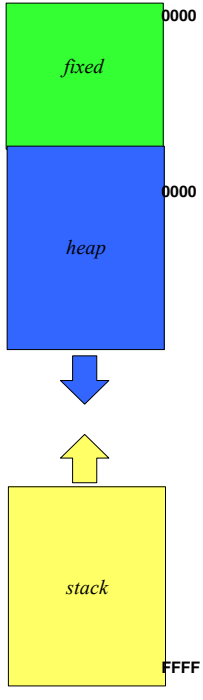
Three memory regions

- When you run a a program, space is allocated from one of several ***memory regions*** depending on the the thing being allocated for.
- One region of memory is reserved for data that is never created or destroyed as the program runs. This is called ***fixed or static memory***.
- One region is reserved for data that needs to be allocated *dynamically*. This is called ***heap memory***.
  - Dynamically allocated memory
    - We don't know how much we need until program *is* running
- On region is reserved for automatic (local variables) defined inside a function. This is called ***stack memory*.**
  - Lives in local memory



+

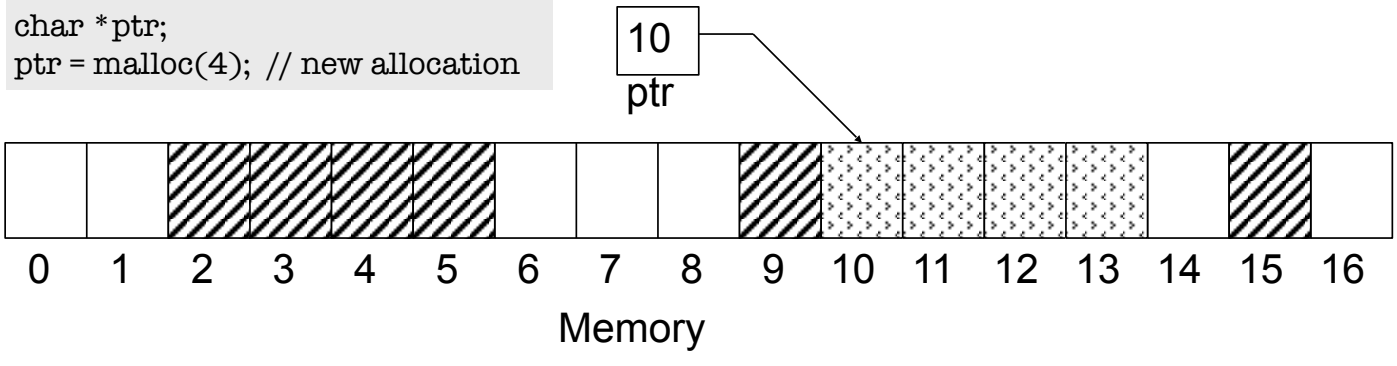
Dynamic Memory Allocation

malloc()

- The malloc() function is used for allocating heap memory at runtime.
- **void* malloc(int size_in_bytes);**
  - searches heap for ‘size’ contiguous free bytes.
  - returns the address of the first byte, unless no memory available then returns the null pointer.
  - programmers responsibility to not lose the pointer.
  - programmers responsibility to respect bounds.
- You must check to make sure that malloc was successful after each allocation!

malloc() example

```
char *ptr;
ptr = malloc(4);  // new allocation
```

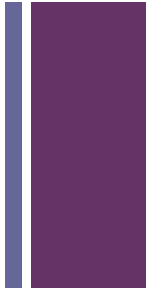


Key

previously allocated

new allocation

C Vs Java



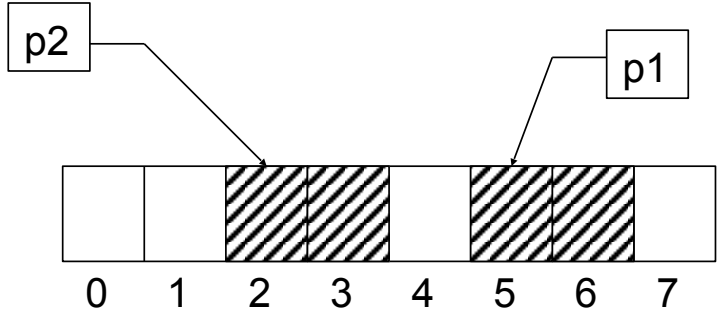
- **malloc()** is a bit like '**new**' in Java.
  - They both allocate space on the heap.
  - They both return the address to the location in the heap where the space requested was allocated.
- There is an important difference though, you do not need to 'clean-up' after yourself in Java.
- In C, you must deallocate memory heap-allocated memory explicitly.

free()

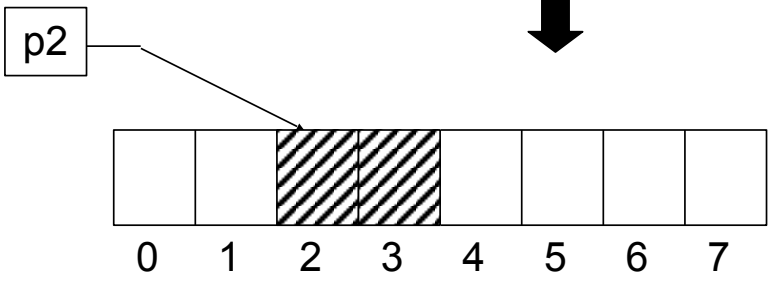
- Any memory allocated with malloc() is reserved, in other words, it can't be used until it is deallocated with free().
- **void free(void* p);**
  - Releases the area pointed to by p.
  - ‘p’ must not be null.
  - System will know how much memory to deallocate.

free() example

```
char *p1;
p1 = malloc(2);
char *p2;
p1 = malloc(2);
```




```
free(p1);
```



Key

allocated memory

free allocation

sizeof()

- The sizeof() function is used to determine the size of any data type
- **int sizeof(type);**
  - returns how many bytes the data type needs
  - for example: sizeof(int) = 4, sizeof(char) = 1
  - works for standard data types and structs
    - after C99, works on variable-length arrays