CS 3

Introduction to Software Design

C, Part 2

poller.com/caltechcs

First poll is open; you can start early (if you want to).

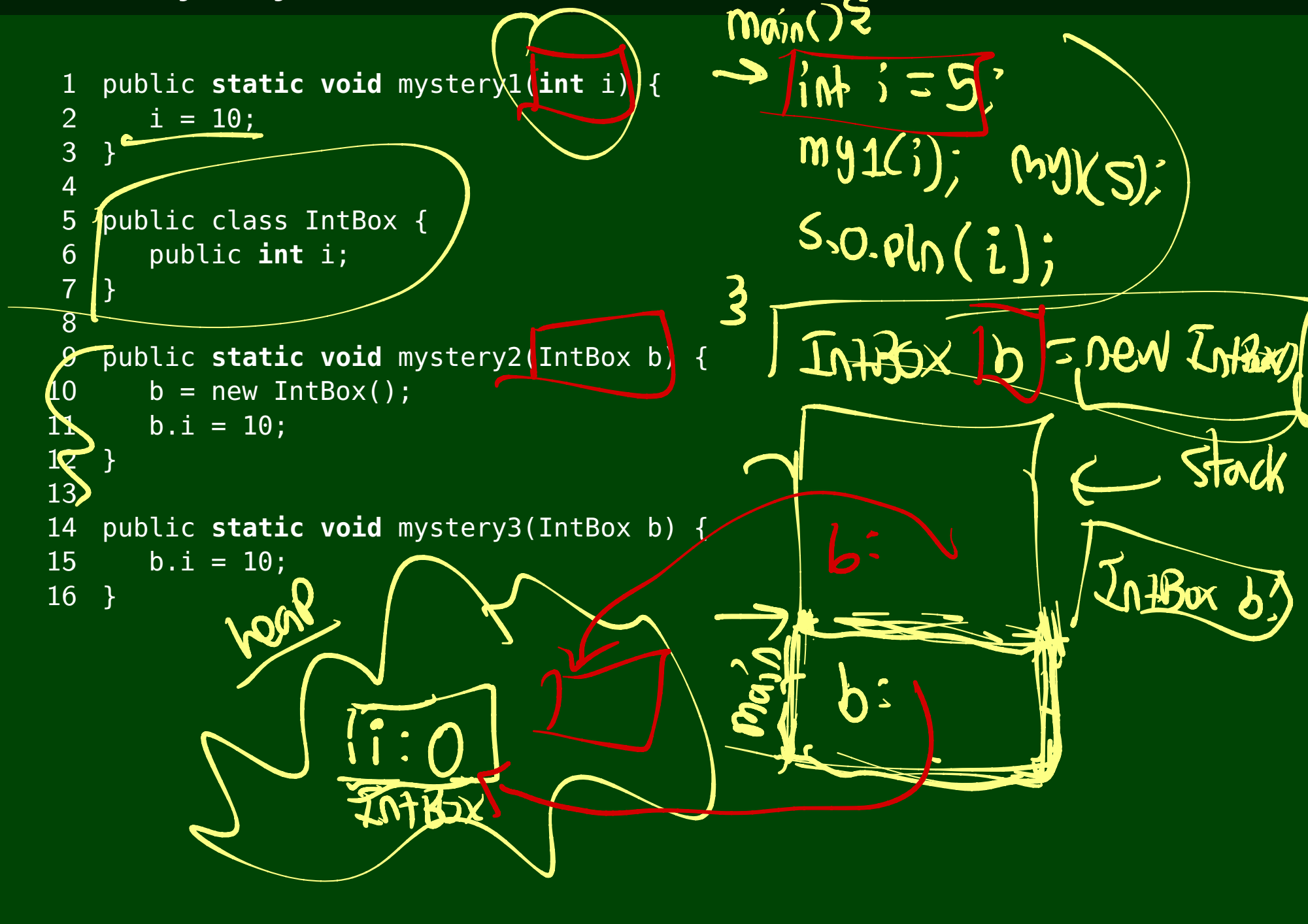
Java Mystery

```
 1  public static void mystery1(int i) {
 2     i = 10;
 3  }
 4
 5  public class IntBox {
 6     public int i;
 7  }
 8
 9  public static void mystery2(IntBox b) {
10     b = new IntBox();
11     b.i = 10;
12  }
13
14  public static void mystery3(IntBox b) {
15     b.i = 10;
16  }
```

DO THE POLL NOW!!

Pollev.com/caltechcs

```
1  public static void mystery1(int i) {
2      i = 10;
3  }
4
5  public class IntBox {
6      public int i;
7  }
8
9  public static void mystery2(IntBox b) {
10     b = new IntBox();
11     b.i = 10;
12 }
13
14 public static void mystery3(IntBox b) {
15     b.i = 10;
16 }
```



```
 1  void cmystery1(int i) {
 2     i = 10;
 3  }
 4
 5  typedef struct int_box {
 6     int i;
 7  } int_box_t;
 8
 9  void cmystery2(int_box_t *b) {
10     b = malloc(sizeof(int_box_t));
11     b->i = 10;
12  }
13
14  void cmystery3(int_box_t *b) {
15     b->i = 10;
16  }
17
18  void cmystery4(int_box_t b) {
19     b.i = 10;
20  }
21
22  void cmystery5(int *i) {
23     *i = 10;
24  }
```

Hints:

(1) sizeof(int) > 1

(2) Does malloc zero memory?

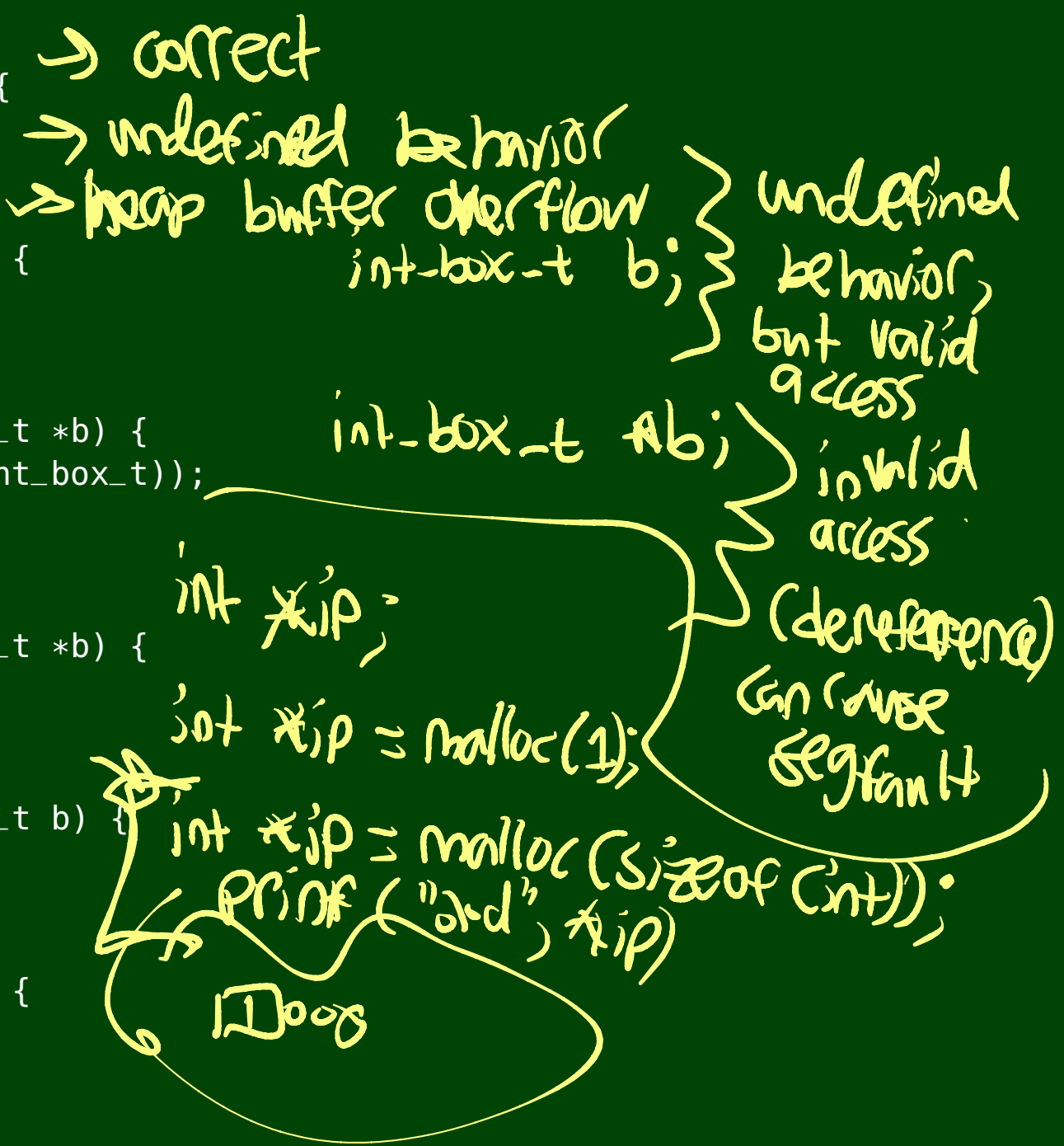
(3) (b->i) ≡ ((*b).i)

int_box_t b;

int *ip;
SEGV "deref of high address"

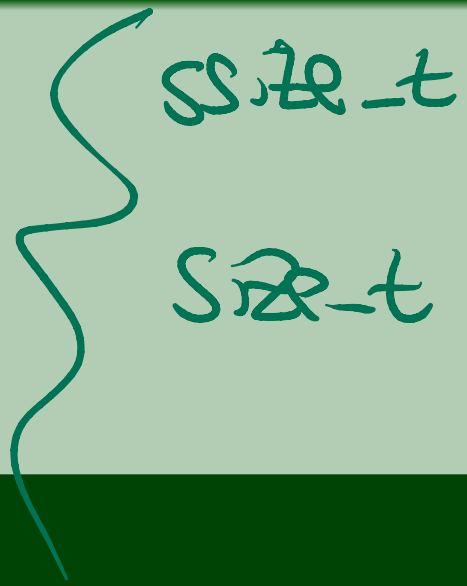
C Mystery

```
 1  void cmystery1(int i) {
 2      i = 10;
 3  }
 4
 5  typedef struct int_box {
 6      int i;
 7  } int_box_t;
 8
 9  void cmystery2(int_box_t *b) {
10      b = malloc(sizeof(int_box_t));
11      b->i = 10;
12  }
13
14  void cmystery3(int_box_t *b) {
15      b->i = 10;
16  }
17
18  void cmystery4(int_box_t b) {
19      b.i = 10;
20  }
21
22  void cmystery5(int *i) {
23      *i = 10;
24  }
```



Pixel in Java

```
1 public class Pixel {
2    public int red;
3    public int green;
4    public int blue;
5
6    public void zeroRed(Pixel p) {
7       p.red = 0;
8    }
9 }
```



pixel_t in C

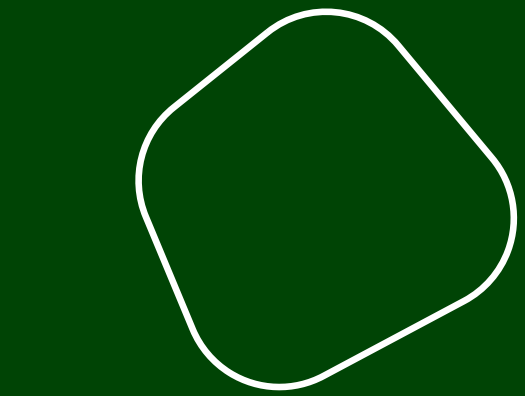
```
1  typedef struct pixel {
2     uint8_t red;
3     uint8_t green;
4     uint8_t blue;
5  } pixel_t;
6
7
8  void pixel_zero_red(pixel_t *p) {
9     p->red = 0;
10 }
```

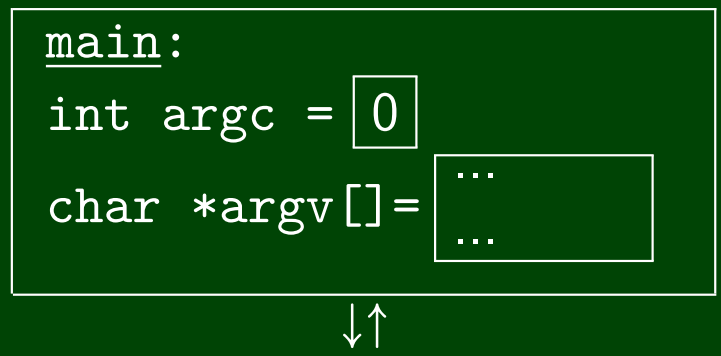
```c
void f(int i) {
    i = 5;
}
```

```c
void g(int *pi) {
    *pi = 5;
}
```

```c
int main(int argc, char *argv[]) {
    int d = 0;
    f(d);
    printf("%d\n", d);

    int *p = malloc(sizeof(int));
    *p = 0;
    g(p);
    printf("%d\n", *p);
    free(p);
}
```



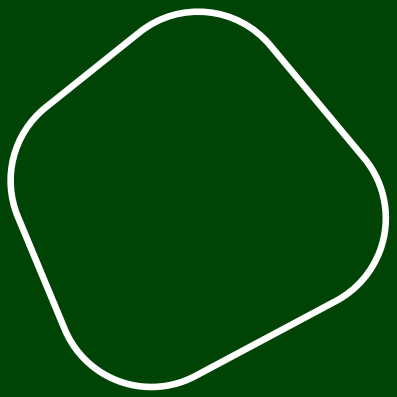


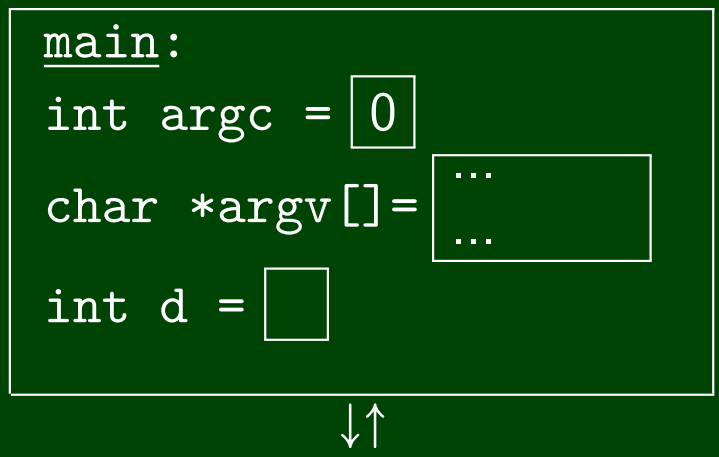
Investigating By Analogy to `int`

```
1  void f(int i) {
2     i = 5;
3  }
```

```
1  void g(int *pi) {
2     *pi = 5;
3  }
```

```
1  int main(int argc, char *argv[]) {
2     int d = 0;
3     f(d);
4     printf("%d\n", d);
5
6     int *p = malloc(sizeof(int));
7     *p = 0;
8     g(p);
9     printf("%d\n", *p);
10    free(p);
11 }
```

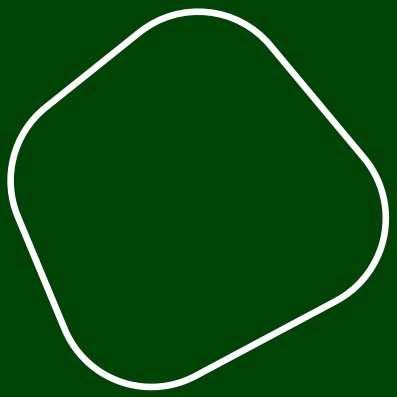


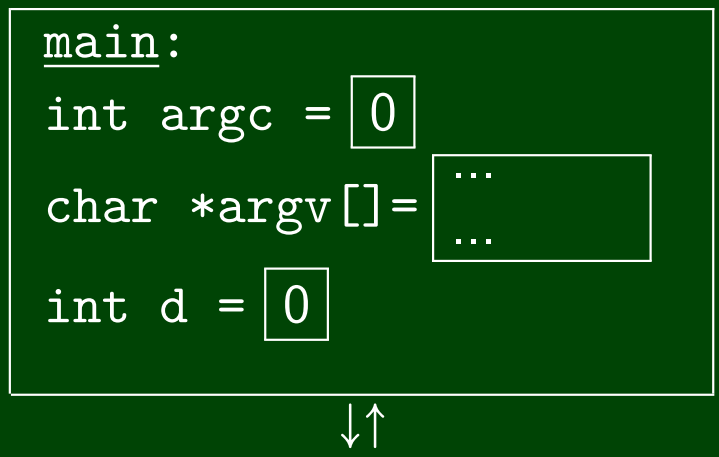


```
1  void f(int i) {
2     i = 5;
3  }
```

```
1  void g(int *pi) {
2     *pi = 5;
3  }
```

```
1  int main(int argc, char *argv[]) {
2     int d = 0;
3     f(d);
4     printf("%d\n", d);
5
6     int *p = malloc(sizeof(int));
7     *p = 0;
8     g(p);
9     printf("%d\n", *p);
10    free(p);
11 }
```



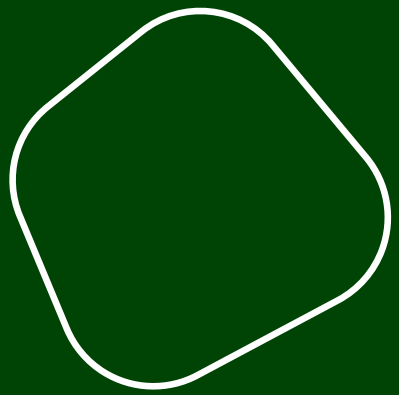


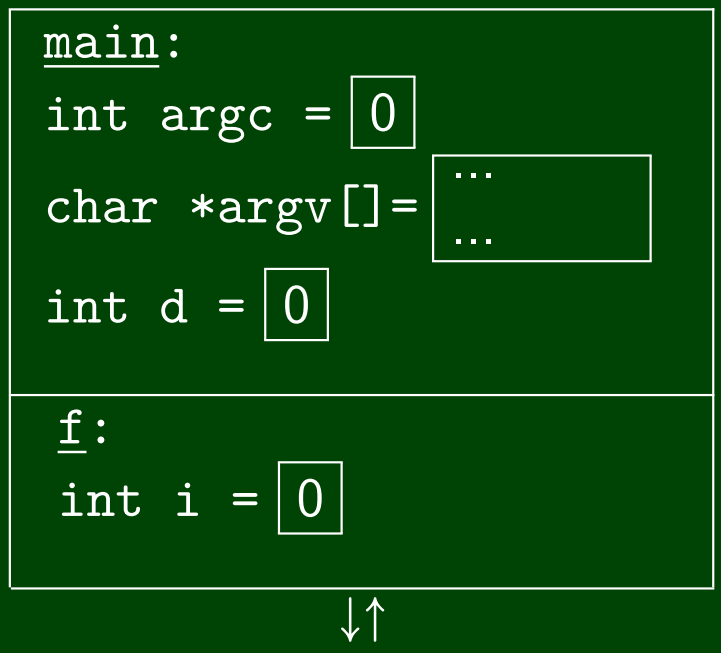
Investigating By Analogy to `int`

```
1  void f(int i) {
2     i = 5;
3  }
```

```
1  void g(int *pi) {
2     *pi = 5;
3  }
```

```
1  int main(int argc, char *argv[]) {
2     int d = 0;
3     f(d);
4     printf("%d\n", d);
5
6     int *p = malloc(sizeof(int));
7     *p = 0;
8     g(p);
9     printf("%d\n", *p);
10    free(p);
11 }
```



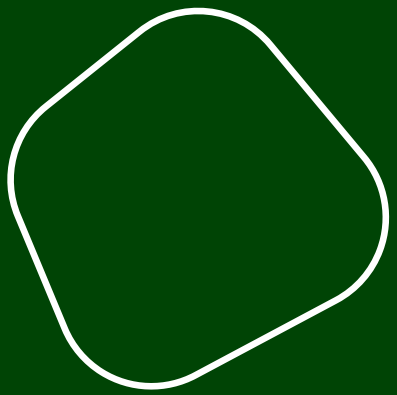


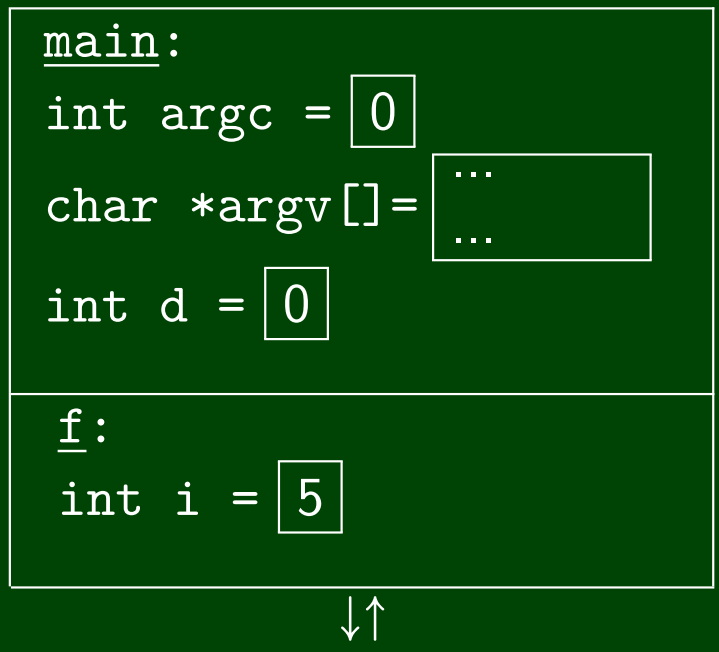
Investigating By Analogy to `int`

```
1  void f(int i) {
2     i = 5;
3  }
```

```
1  void g(int *pi) {
2     *pi = 5;
3  }
```

```
1  int main(int argc, char *argv[]) {
2     int d = 0;
3     f(d);
4     printf("%d\n", d);
5
6     int *p = malloc(sizeof(int));
7     *p = 0;
8     g(p);
9     printf("%d\n", *p);
10    free(p);
11 }
```



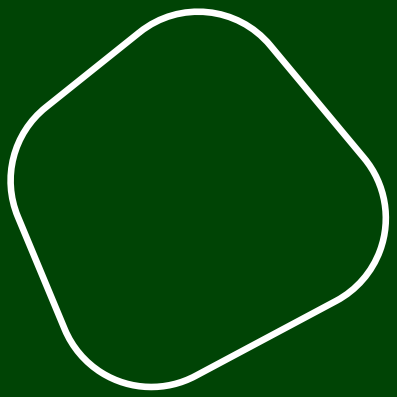


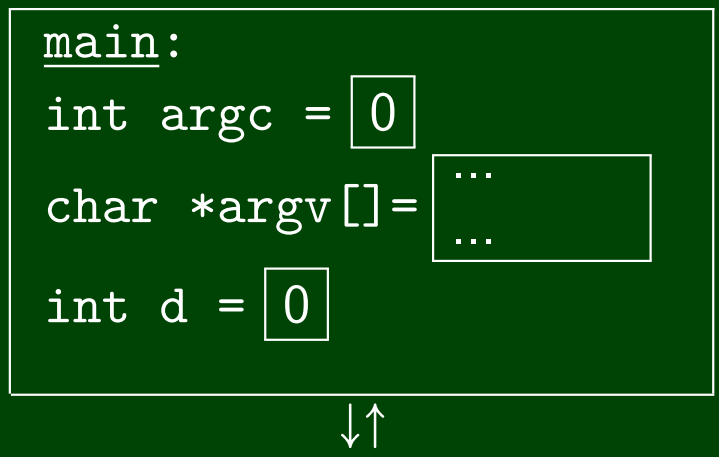
Investigating By Analogy to `int`

```
1  void f(int i) {
2     i = 5;
3  }
```

```
1  void g(int *pi) {
2     *pi = 5;
3  }
```

```
1  int main(int argc, char *argv[]) {
2     int d = 0;
3     f(d);
4     printf("%d\n", d);
5
6     int *p = malloc(sizeof(int));
7     *p = 0;
8     g(p);
9     printf("%d\n", *p);
10    free(p);
11 }
```



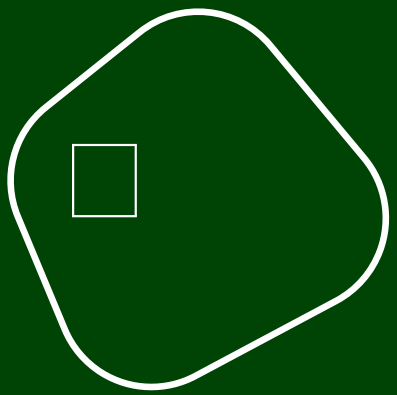


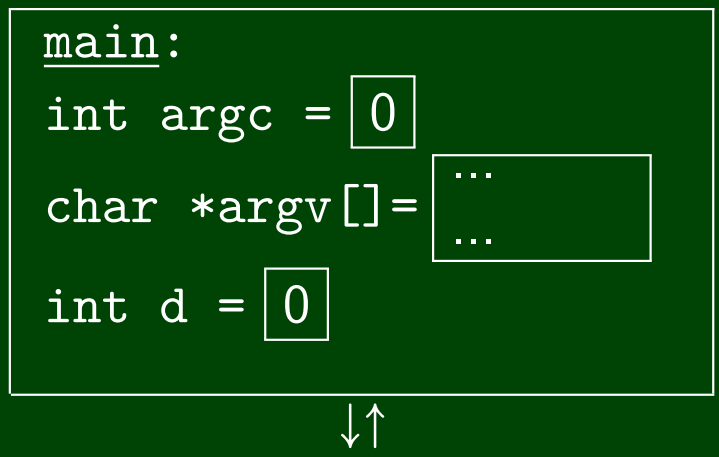
Investigating By Analogy to `int`

```
1  void f(int i) {
2     i = 5;
3  }
```

```
1  void g(int *pi) {
2     *pi = 5;
3  }
```

```
1  int main(int argc, char *argv[]) {
2     int d = 0;
3     f(d);
4     printf("%d\n", d);
5
6     int *p = malloc(sizeof(int));
7     *p = 0;
8     g(p);
9     printf("%d\n", *p);
10    free(p);
11 }
```



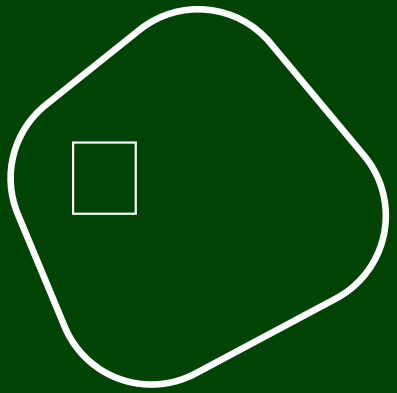


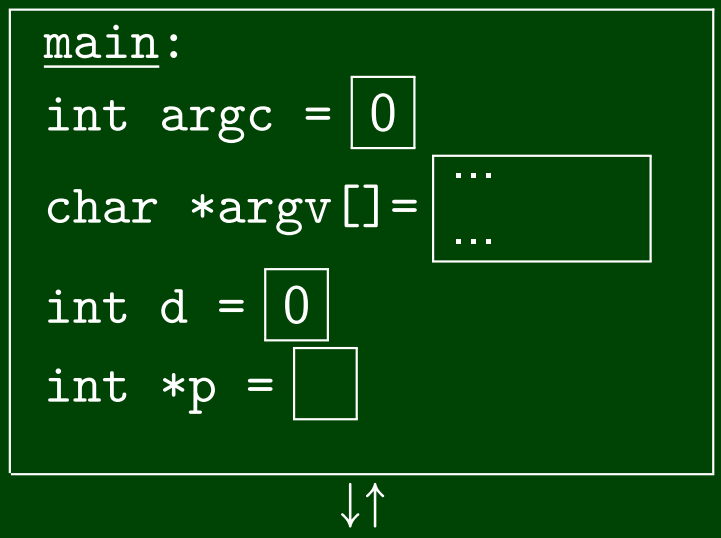
Investigating By Analogy to `int`


```
1  void f(int i) {
2     i = 5;
3  }
```

```
1  void g(int *pi) {
2     *pi = 5;
3  }
```

```
1  int main(int argc, char *argv[]) {
2     int d = 0;
3     f(d);
4     printf("%d\n", d);
5
6     int *p = malloc(sizeof(int));
7     *p = 0;
8     g(p);
9     printf("%d\n", *p);
10    free(p);
11 }
```



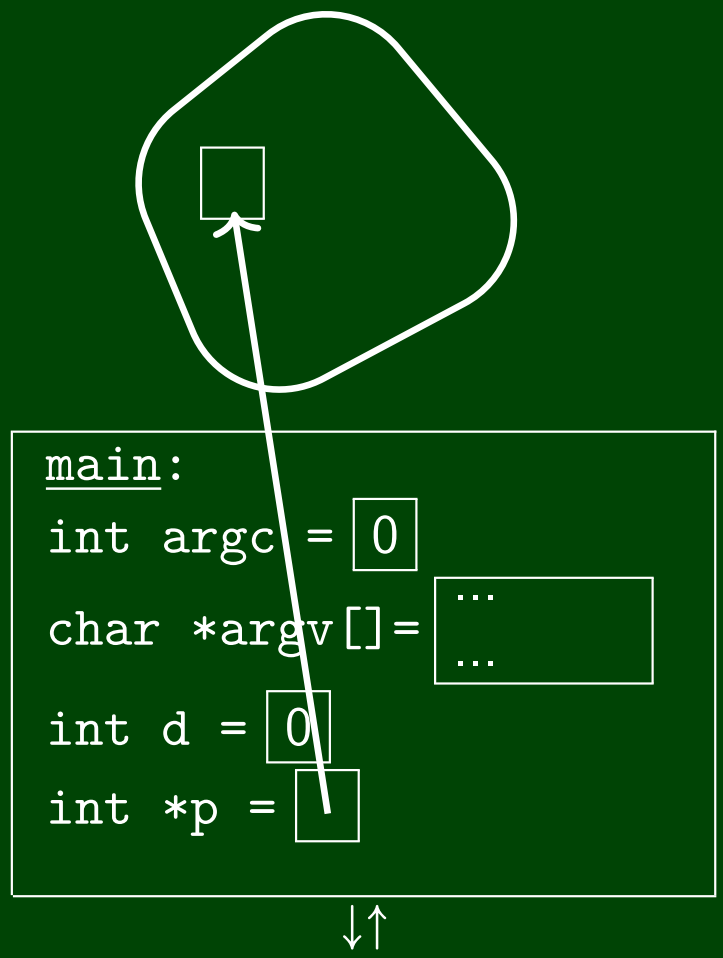


Investigating By Analogy to `int`


```
1  void f(int i) {
2     i = 5;
3  }
```

```
1  void g(int *pi) {
2     *pi = 5;
3  }
```

```
1  int main(int argc, char *argv[]) {
2     int d = 0;
3     f(d);
4     printf("%d\n", d);
5
6     int *p = malloc(sizeof(int));
7     *p = 0;
8     g(p);
9     printf("%d\n", *p);
10    free(p);
11 }
```

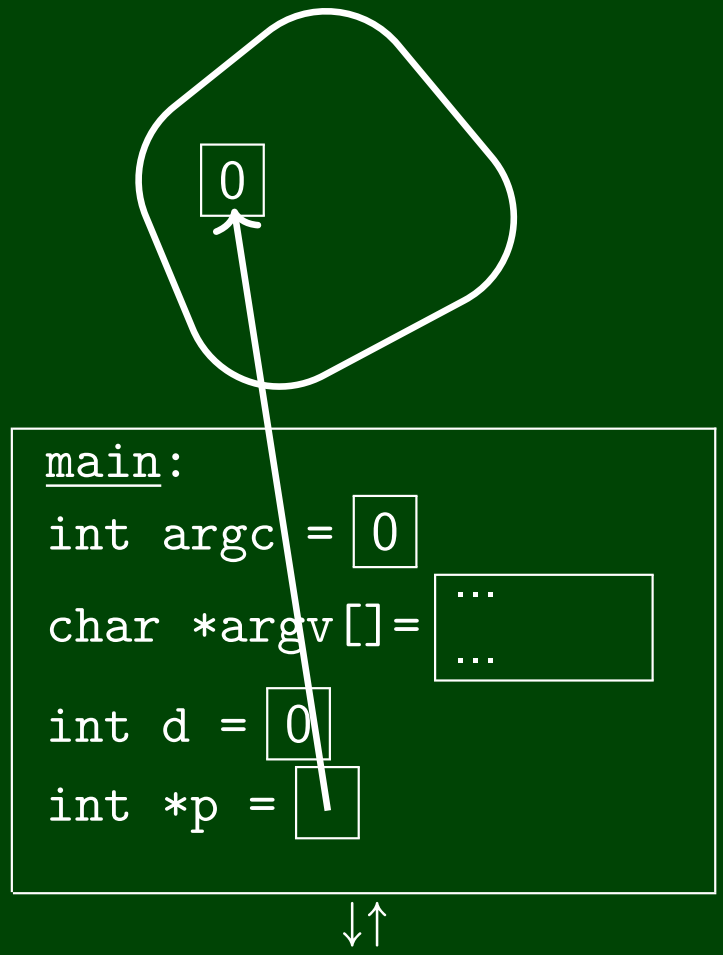


Investigating By Analogy to `int`

```
1  void f(int i) {
2     i = 5;
3  }
```

```
1  void g(int *pi) {
2     *pi = 5;
3  }
```

```
1  int main(int argc, char *argv[]) {
2     int d = 0;
3     f(d);
4     printf("%d\n", d);
5
6     int *p = malloc(sizeof(int));
7     *p = 0;
8     g(p);
9     printf("%d\n", *p);
10    free(p);
11 }
```

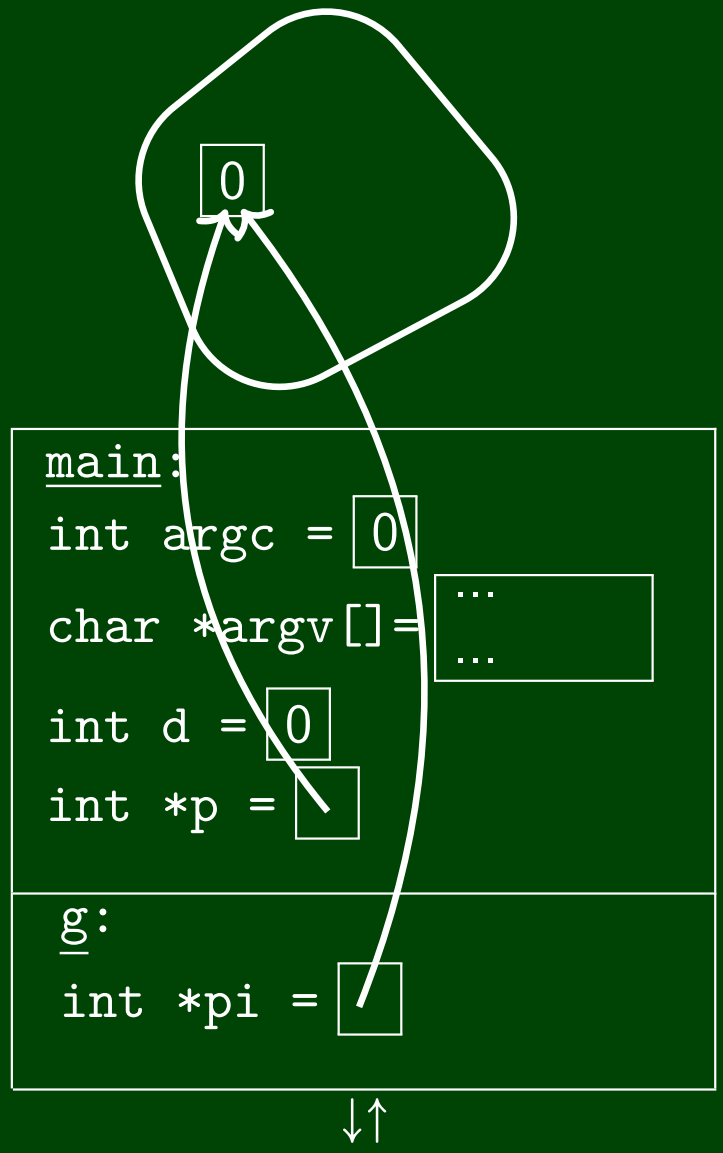


Investigating By Analogy to `int`


```
1  void f(int i) {
2     i = 5;
3  }
```

```
1  void g(int *pi) {
2     *pi = 5;
3  }
```

```
1  int main(int argc, char *argv[]) {
2     int d = 0;
3     f(d);
4     printf("%d\n", d);
5
6     int *p = malloc(sizeof(int));
7     *p = 0;
8     g(p);
9     printf("%d\n", *p);
10    free(p);
11 }
```

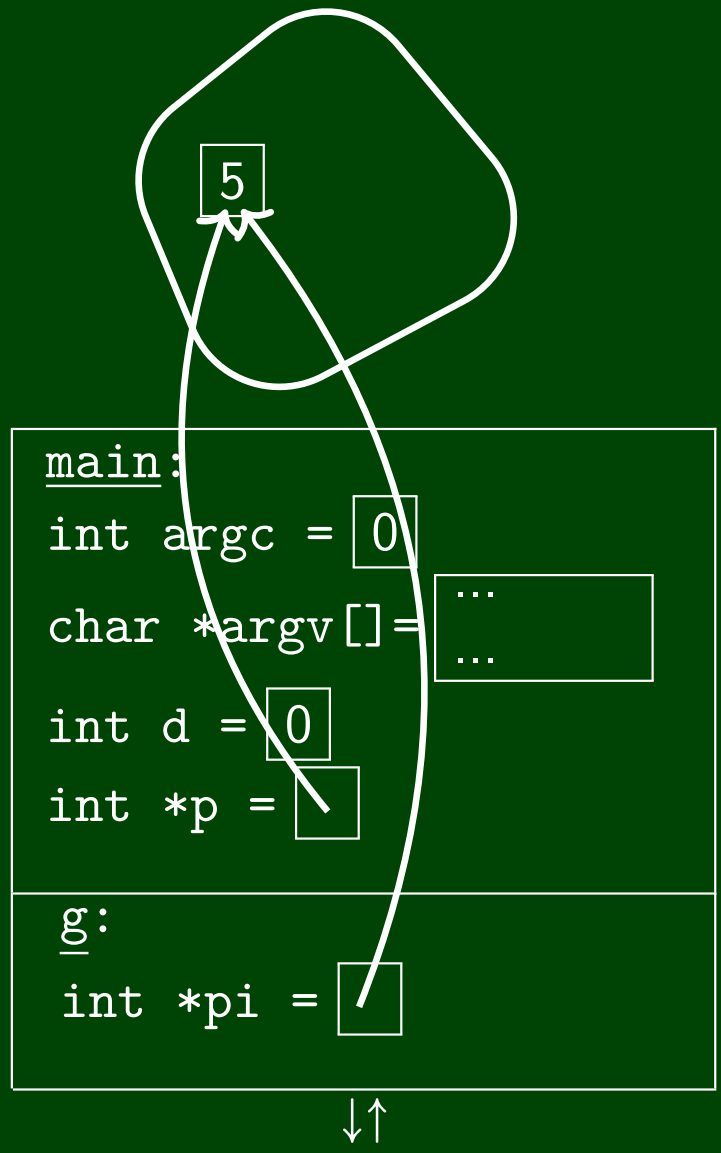


Investigating By Analogy to `int`

```
1  void f(int i) {
2     i = 5;
3  }
```

```
1  void g(int *pi) {
2     *pi = 5;
3  }
```

```
1  int main(int argc, char *argv[]) {
2     int d = 0;
3     f(d);
4     printf("%d\n", d);
5
6     int *p = malloc(sizeof(int));
7     *p = 0;
8     g(p);
9     printf("%d\n", *p);
10    free(p);
11 }
```

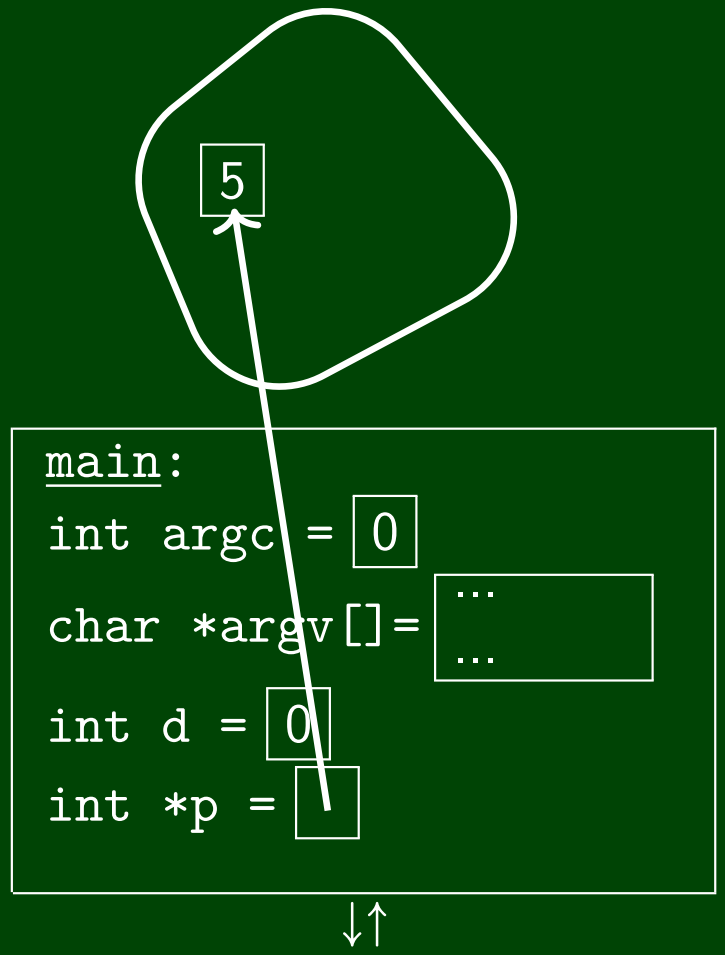


Investigating By Analogy to `int`


```
1  void f(int i) {
2     i = 5;
3  }
```

```
1  void g(int *pi) {
2     *pi = 5;
3  }
```

```
1  int main(int argc, char *argv[]) {
2     int d = 0;
3     f(d);
4     printf("%d\n", d);
5
6     int *p = malloc(sizeof(int));
7     *p = 0;
8     g(p);
9     printf("%d\n", *p);
10    free(p);
11 }
```

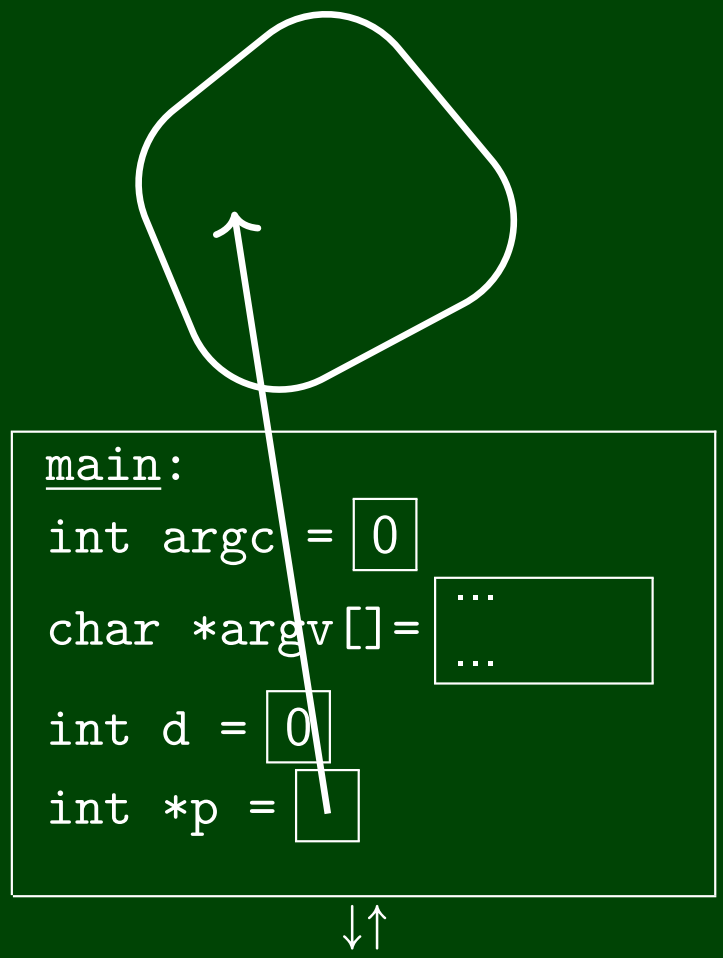


Investigating By Analogy to `int`

```
1  void f(int i) {
2     i = 5;
3  }
```

```
1  void g(int *pi) {
2     *pi = 5;
3  }
```

```
1  int main(int argc, char *argv[]) {
2     int d = 0;
3     f(d);
4     printf("%d\n", d);
5
6     int *p = malloc(sizeof(int));
7     *p = 0;
8     g(p);
9     printf("%d\n", *p);
10    free(p);
11 }
```



Allocating on the Stack

```
 1 // Initializes pixel on the stack
 2 pixel_t p;
 3 p.red = 0;
 4 p.green = 255;
 5 p.blue = 0;
 6
 7 // Shorthand for initializing pixel on the stack
 8 pixel_t p2 = {
 9     .red = 0,
10     .green = 255,
11     .blue = 0
12 };
```