Goals for Today

Implement generic functions

- Passing/returning pointers to elements
- Using client-provided callback functions

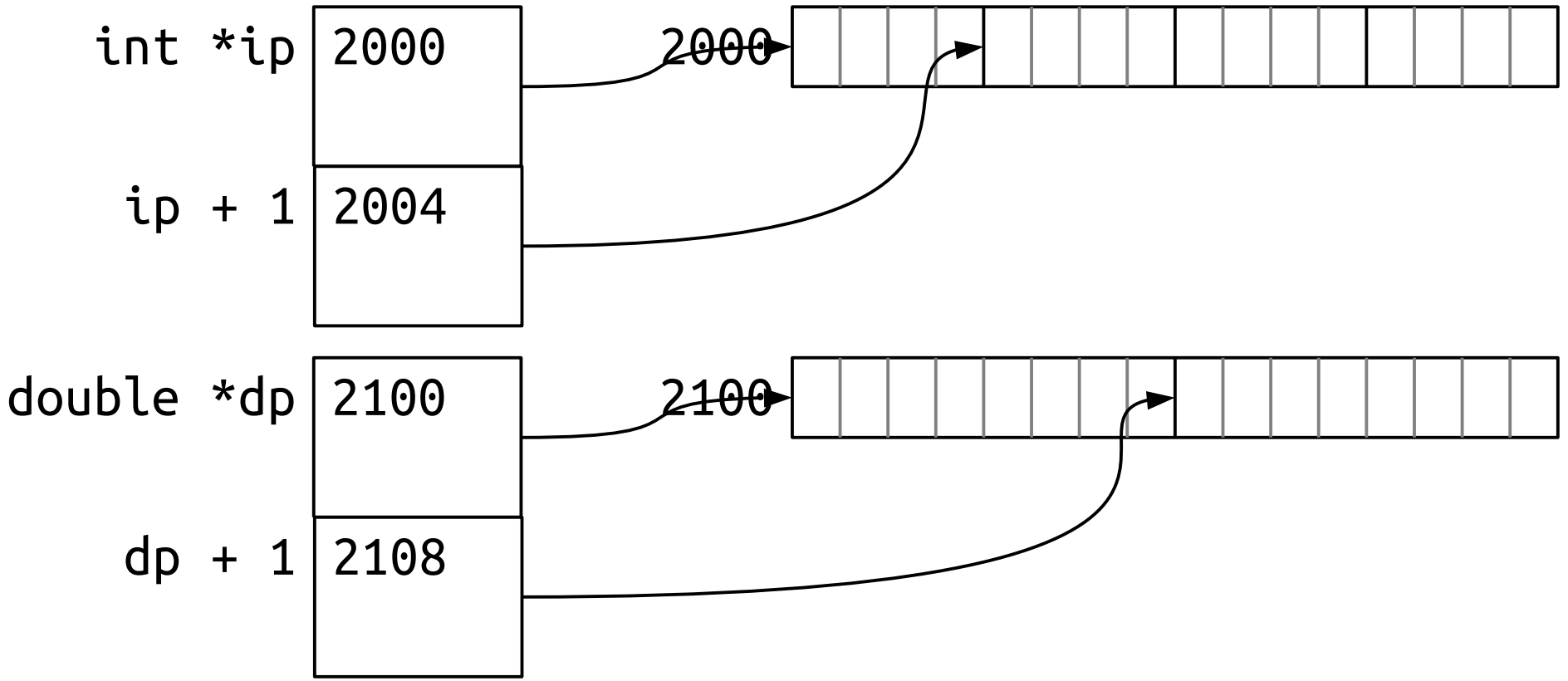
Discuss limitations of `void *`

- Cannot use pointer arithmetic on `void *`
- Cannot dereference `void *`

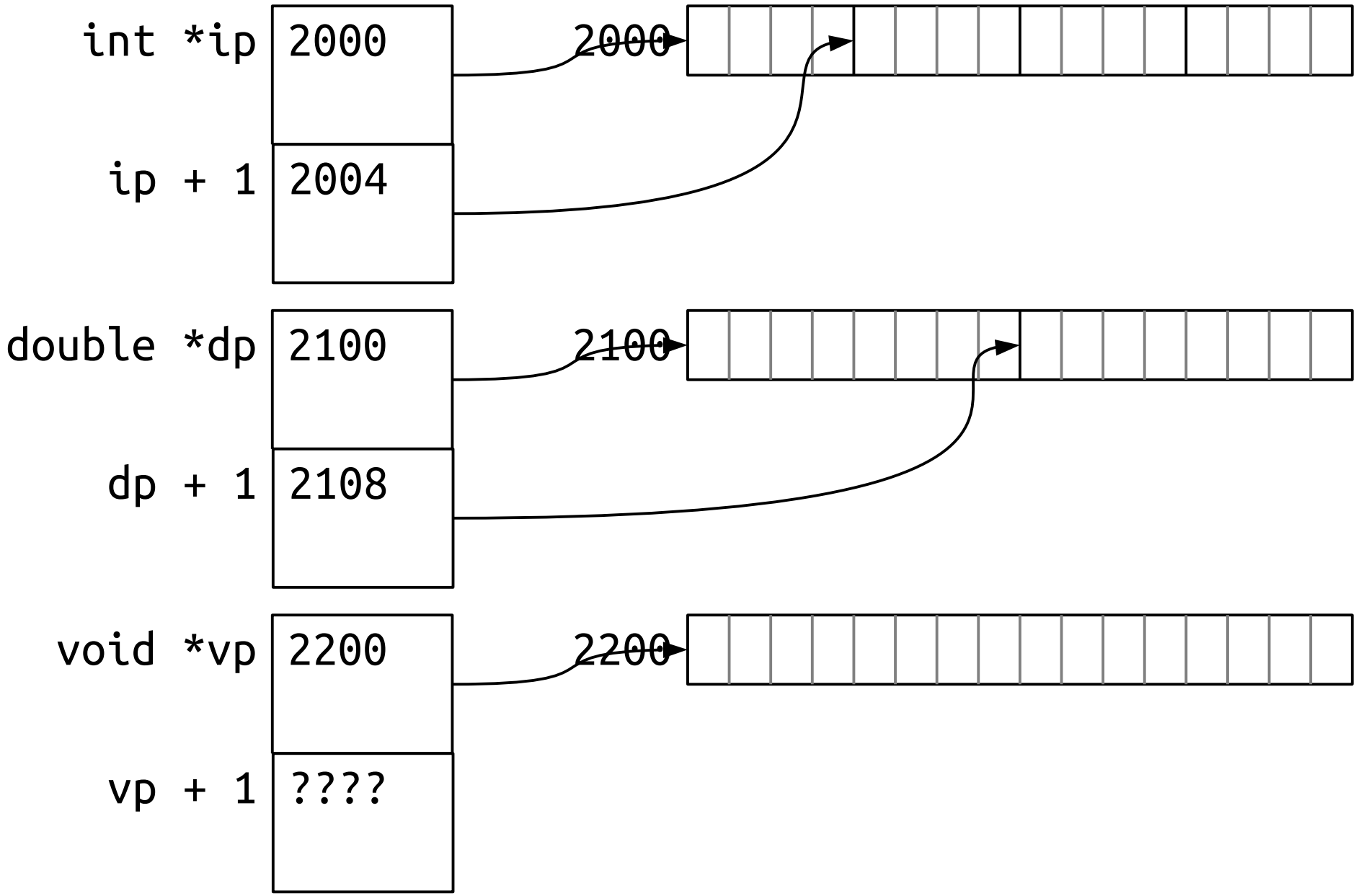
Explore level of indirection errors

Implement find_max

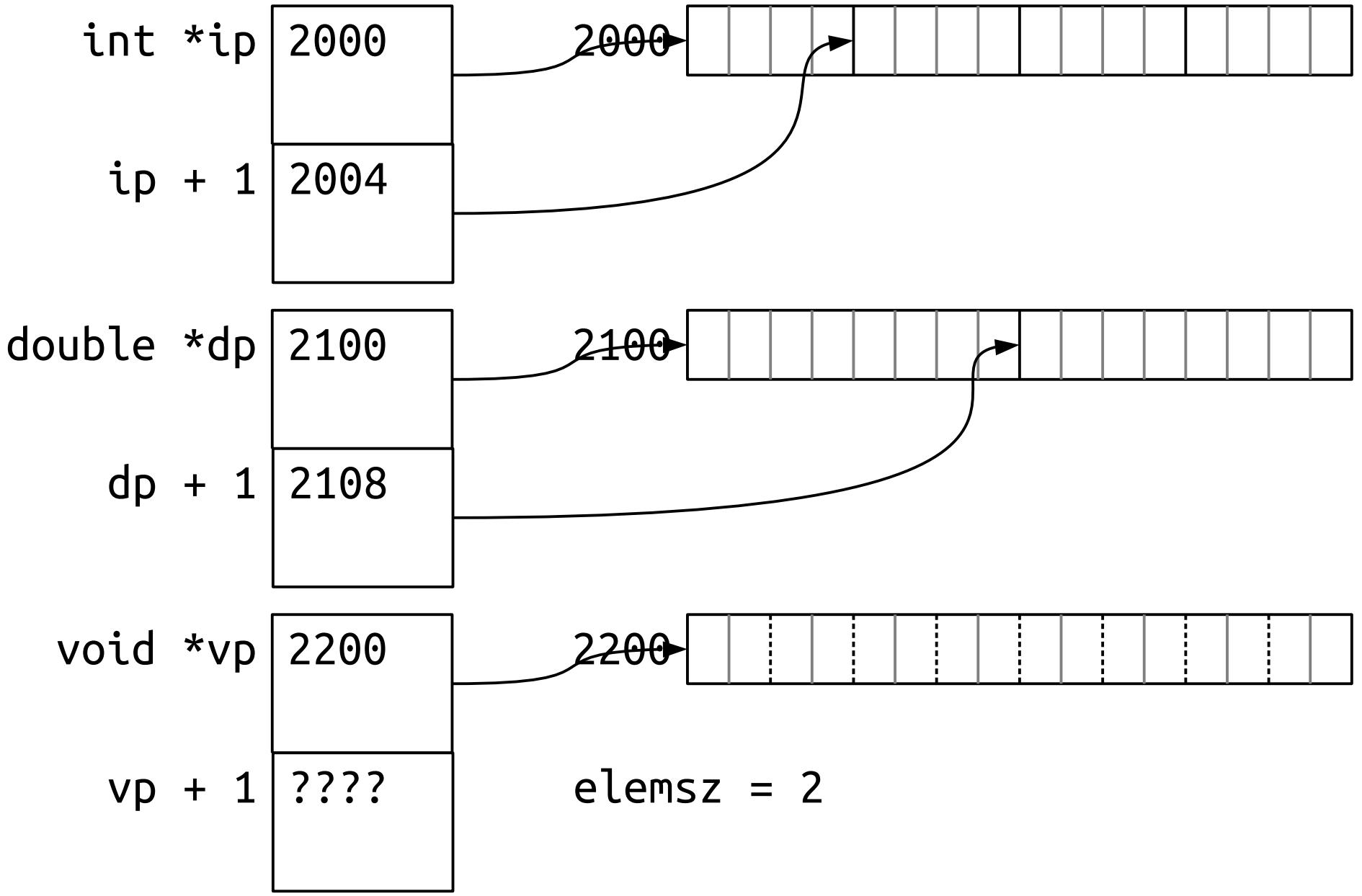
Pointer Arithmetic with void *



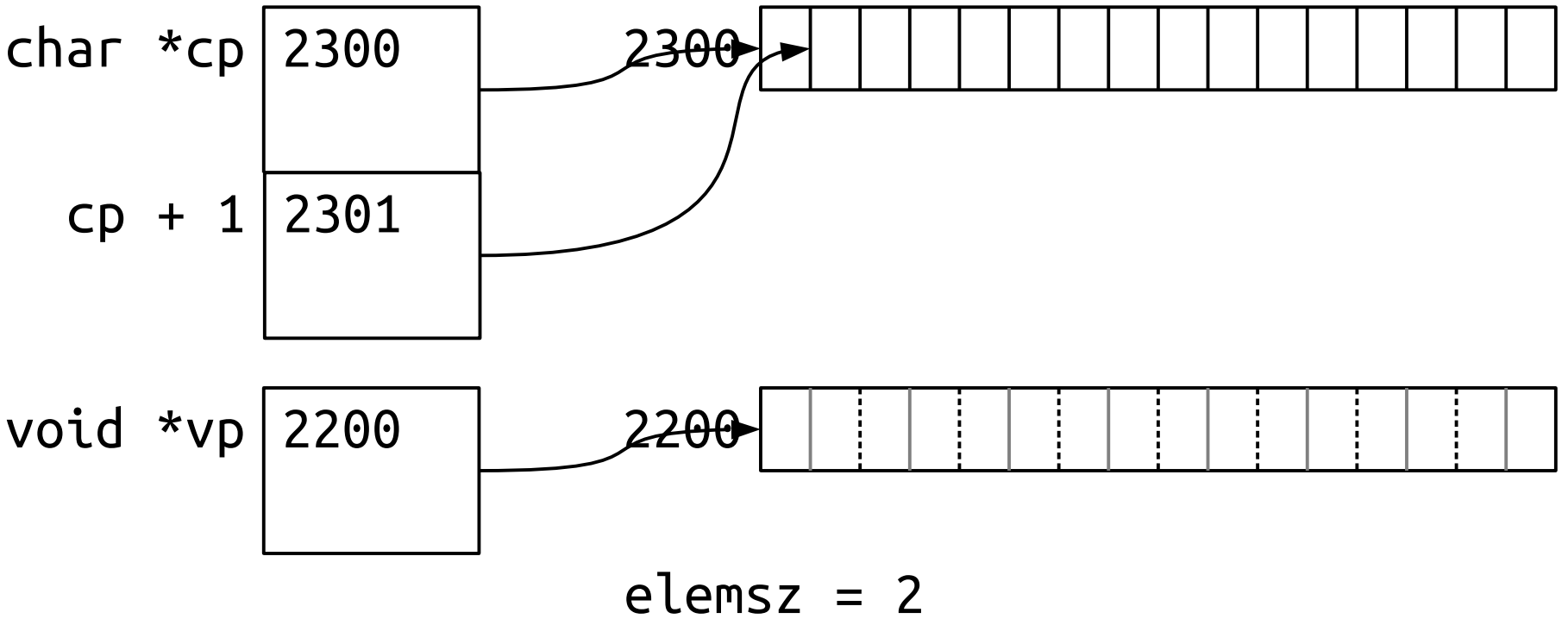
Pointer Arithmetic with void *



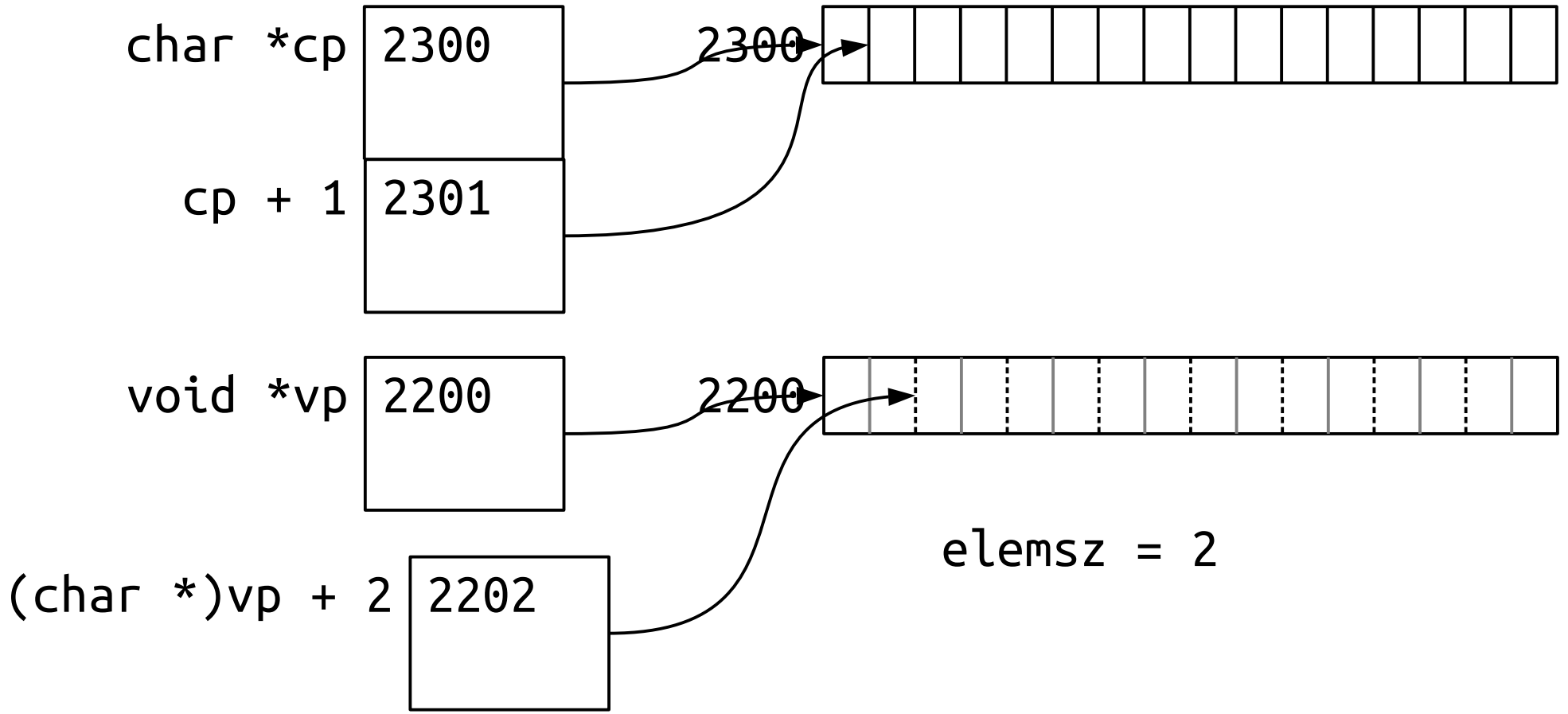
Pointer Arithmetic with void *



Pointer Arithmetic with void *



Pointer Arithmetic with void *



Treat `void *` as `char *` for pointer arithmetic

Pointers to Elements

Generic functions operate on pointers to elements

If array elements have type	... Pointer to element has type
`int`	`int *`
`double`	`double *`
`char`	`char *`
`char *`	`char **`

So Far

Implement generic functions

- Passing/returning pointers to elements
- Using client-provided callback functions

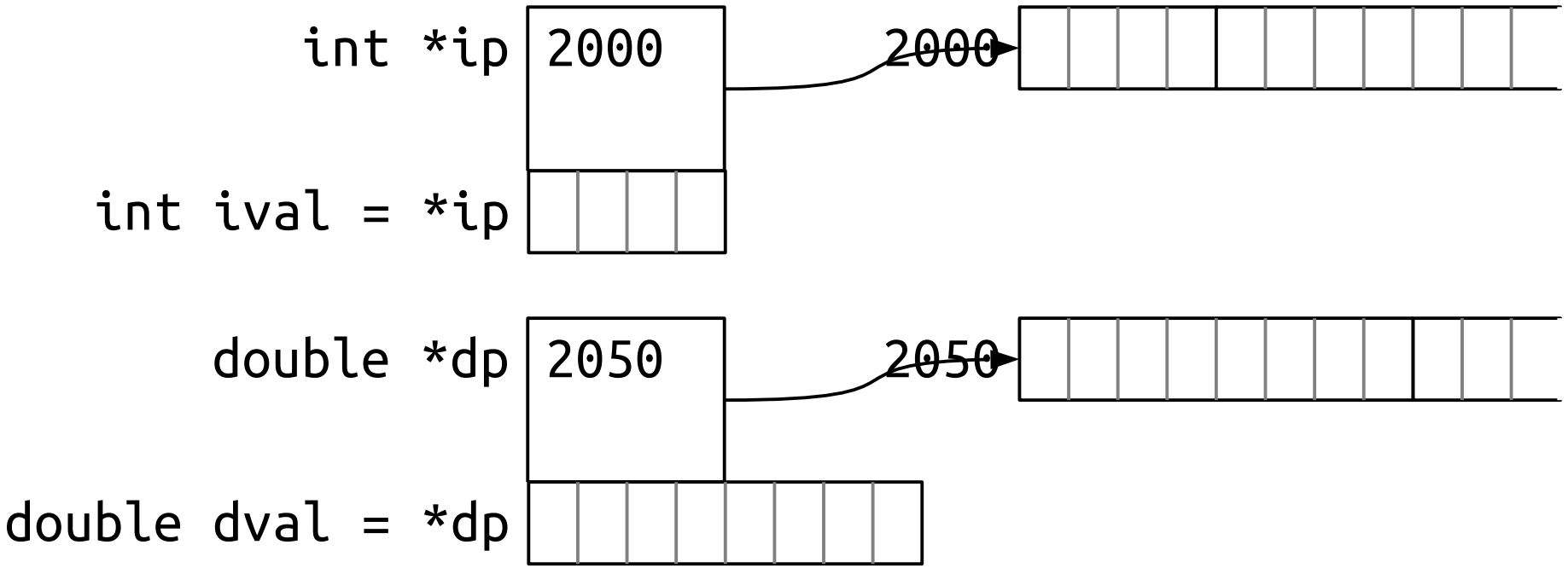
Discuss limitations of `void *`

- Cannot use pointer arithmetic on `void *`
- Cannot dereference `void *`

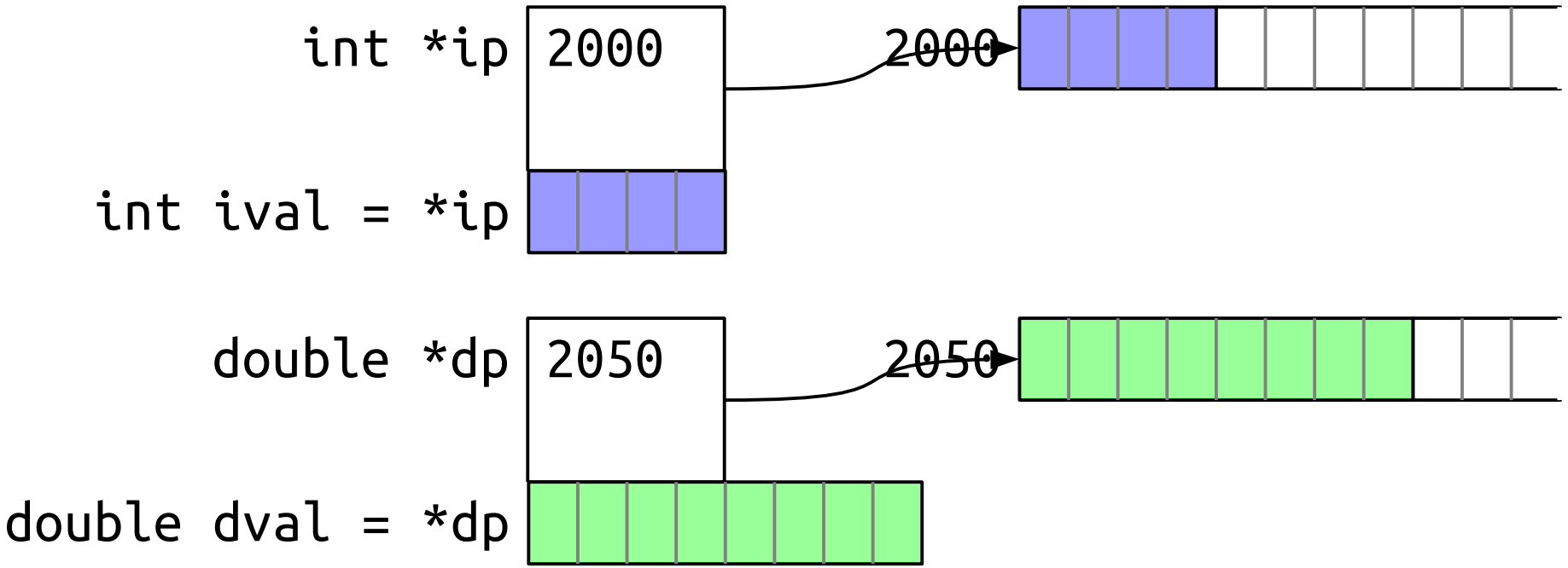
Explore level of indirection errors

Implement gswap

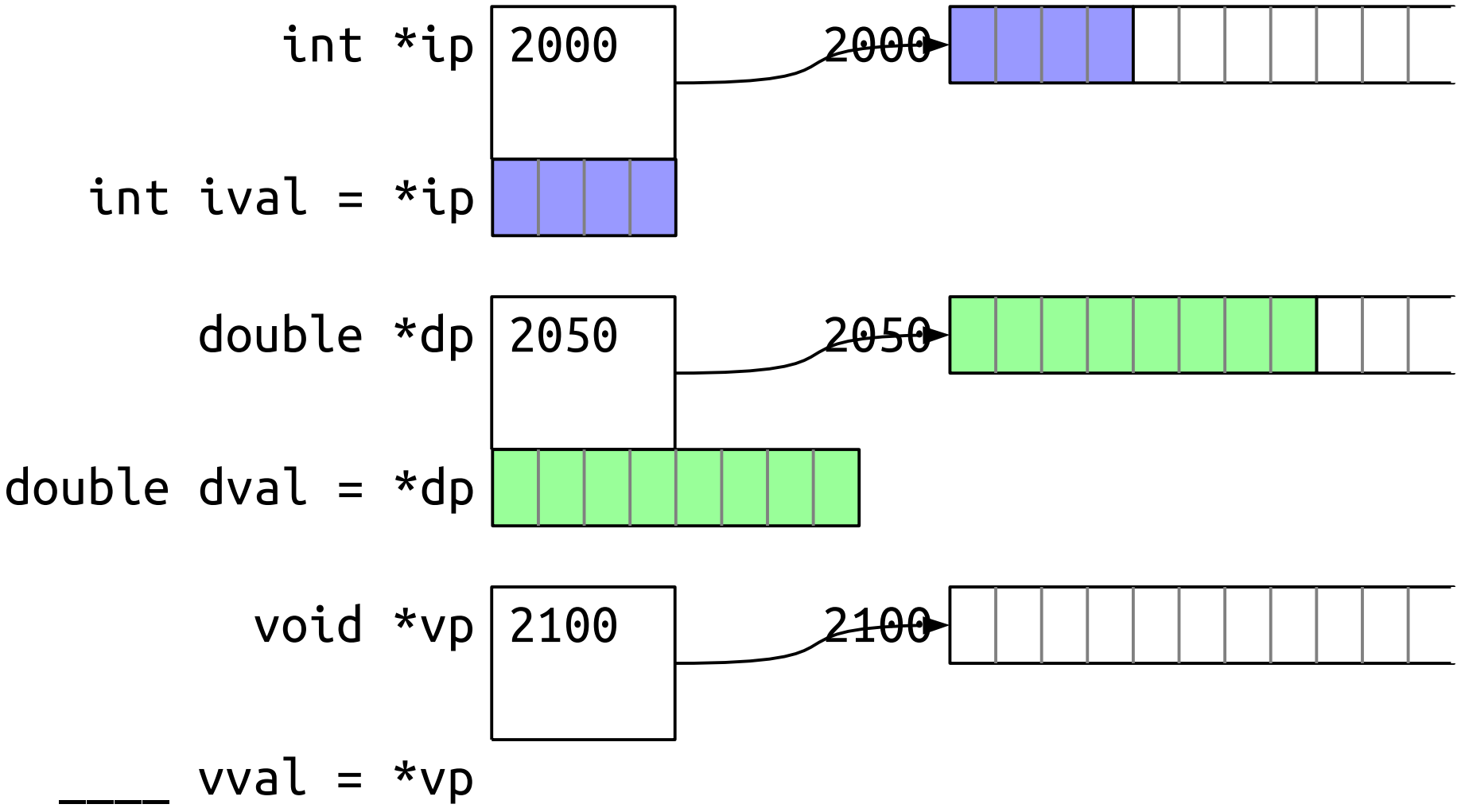
Dereferencing void *



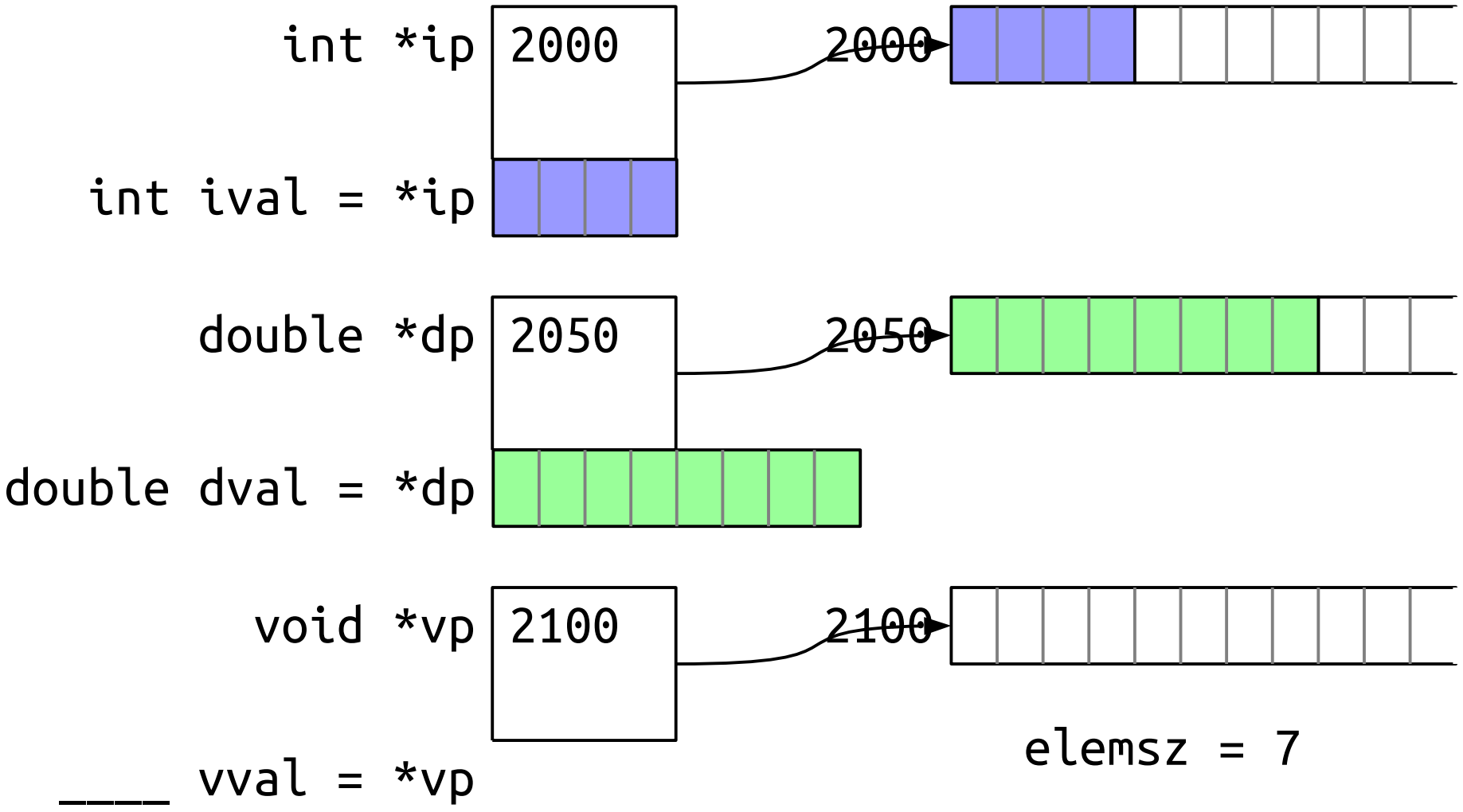
Dereferencing void *



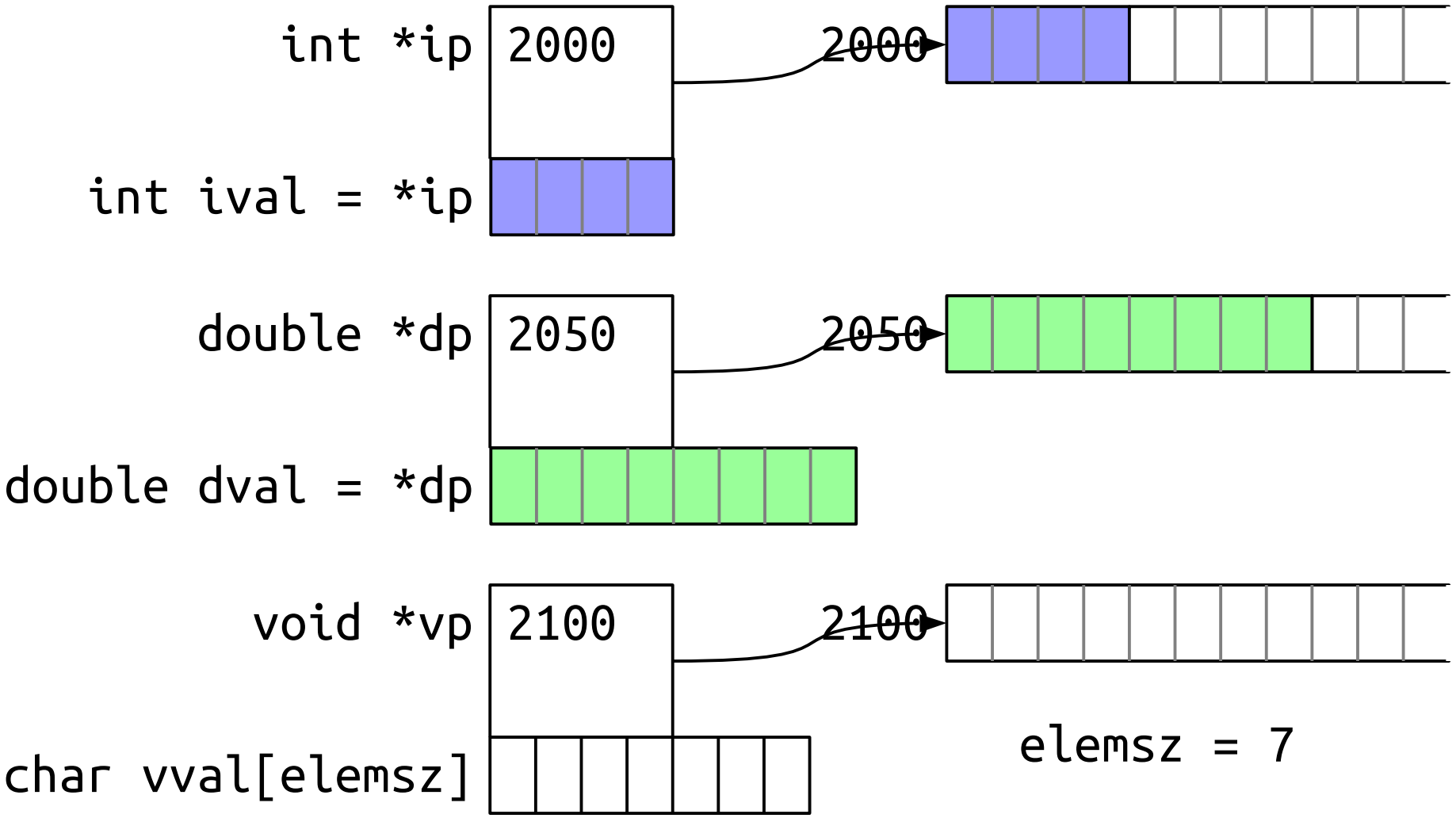
Dereferencing void *



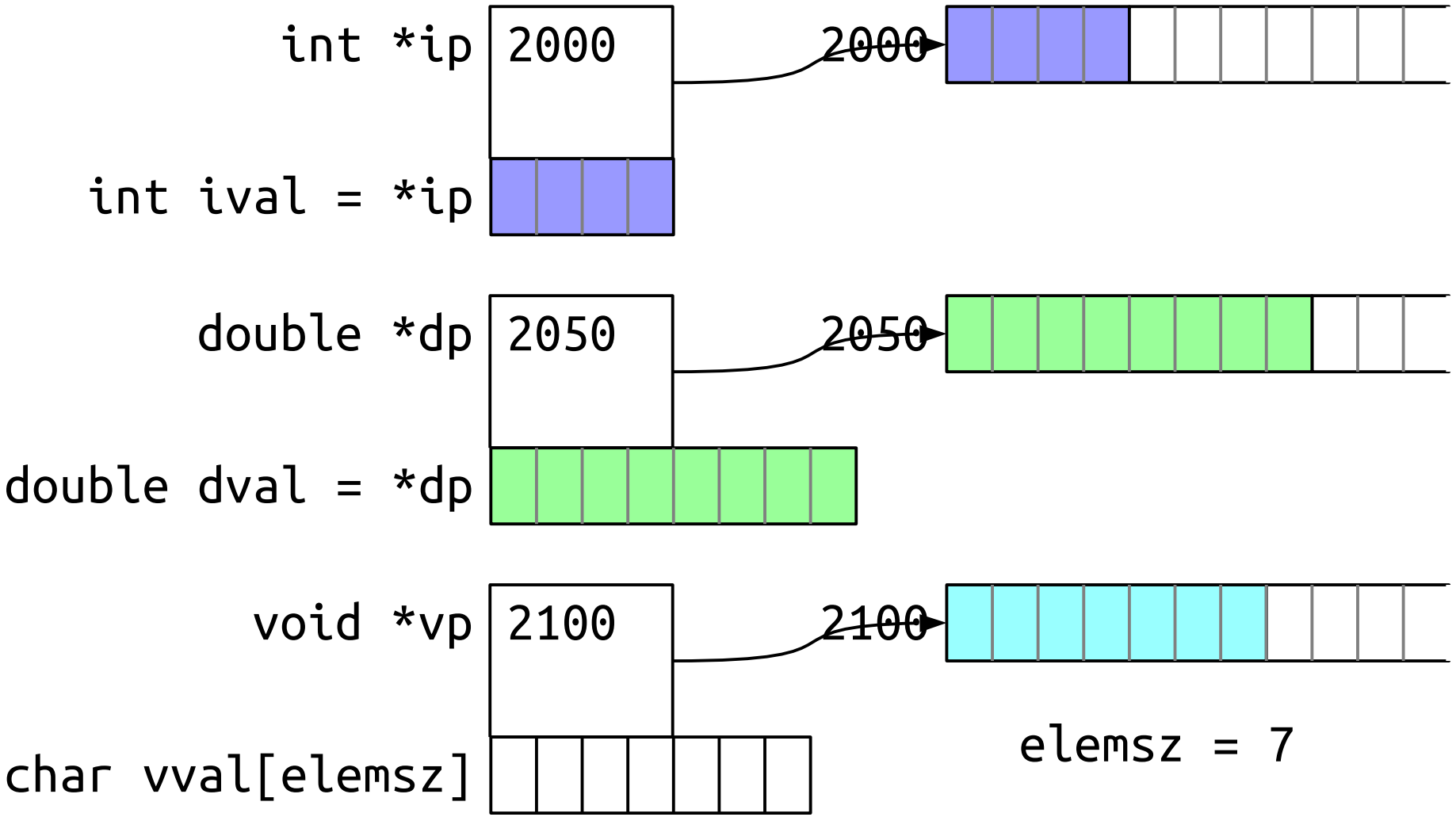
Dereferencing void *



Dereferencing void *

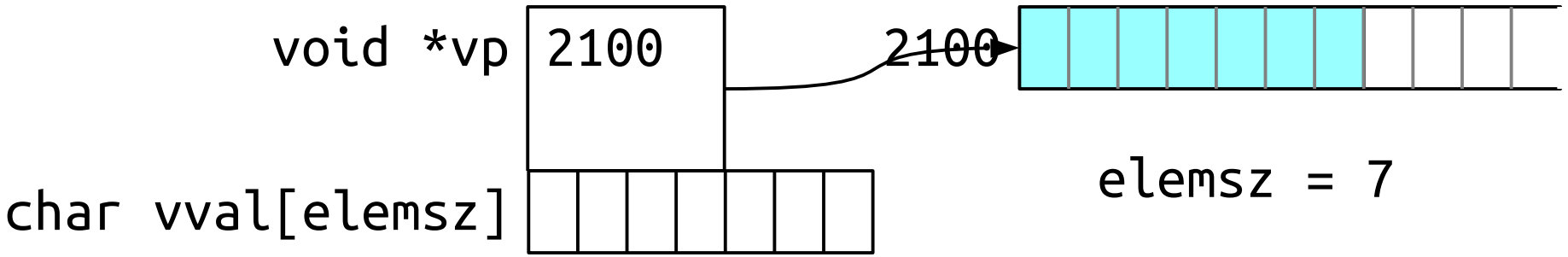


Dereferencing void *



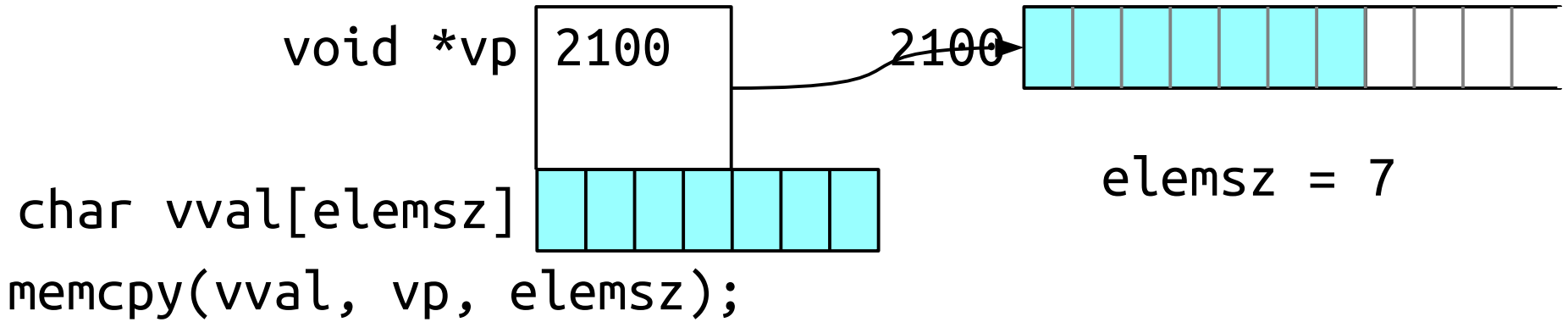
Dereferencing void *

```
memcpy(void *dst, void *src, size_t sz);
```



Dereferencing void *

```
memcpy(void *dst, void *src, size_t sz);
```



So Far

Implement generic functions

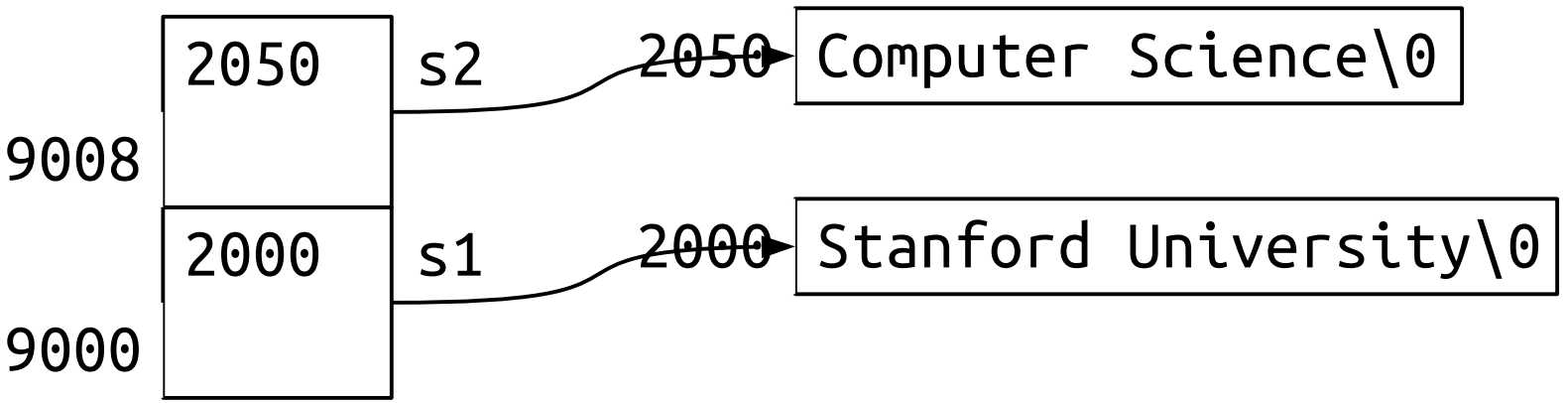
- Passing/returning pointers to elements
- Using client-provided callback functions

Discuss limitations of `void *`

- Cannot use pointer arithmetic on `void *`
- Cannot dereference `void *`

Explore level of indirection errors

Using gswap

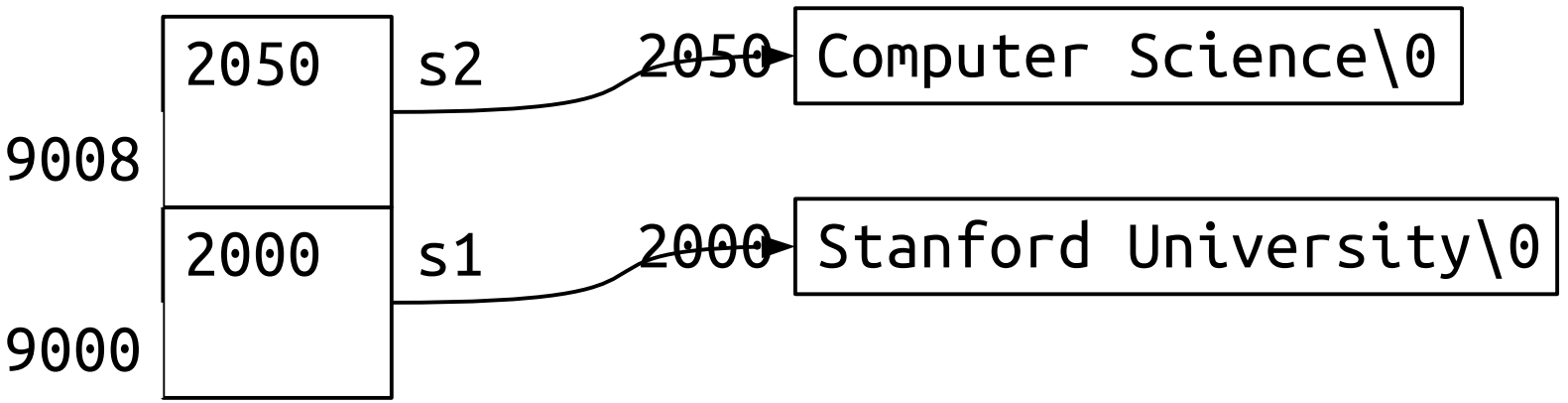



```
char *s1 = strdup("Stanford University");
char *s2 = strdup("Computer Science");

/* Which call is correct? */
gswap(&s1, &s2, sizeof(char *)); // (1)
gswap(s1, s2, sizeof(char *)); // (2)

free(s1);
free(s2);
```

Using gswap

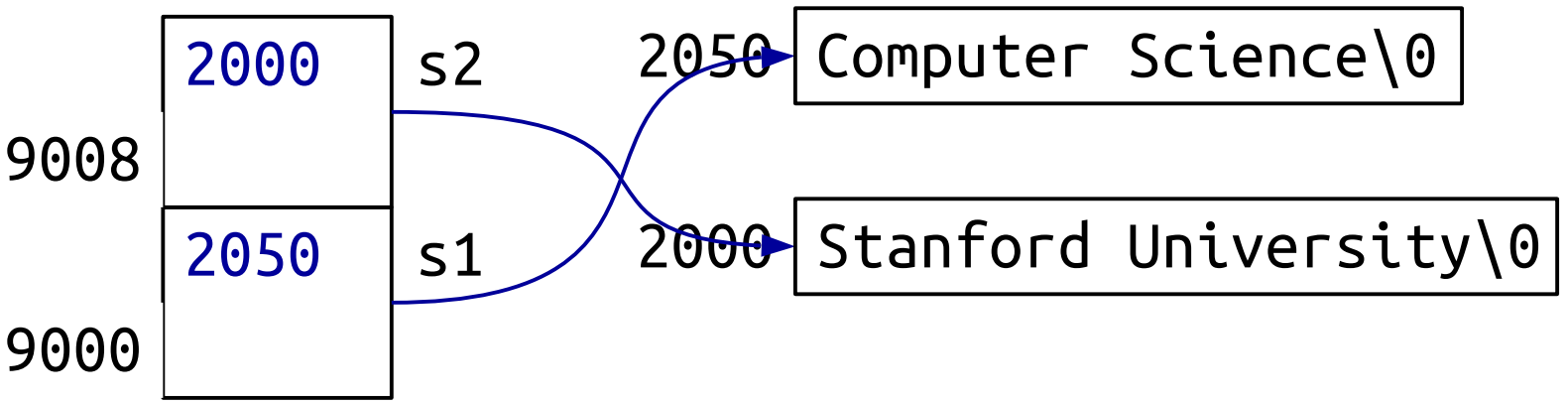



```
char *s1 = strdup(“Stanford University”);
char *s2 = strdup(“Computer Science”);


gswap(&s1, &s2, sizeof(char *)); // (1)


free(s1);
free(s2);
```

Using gswap

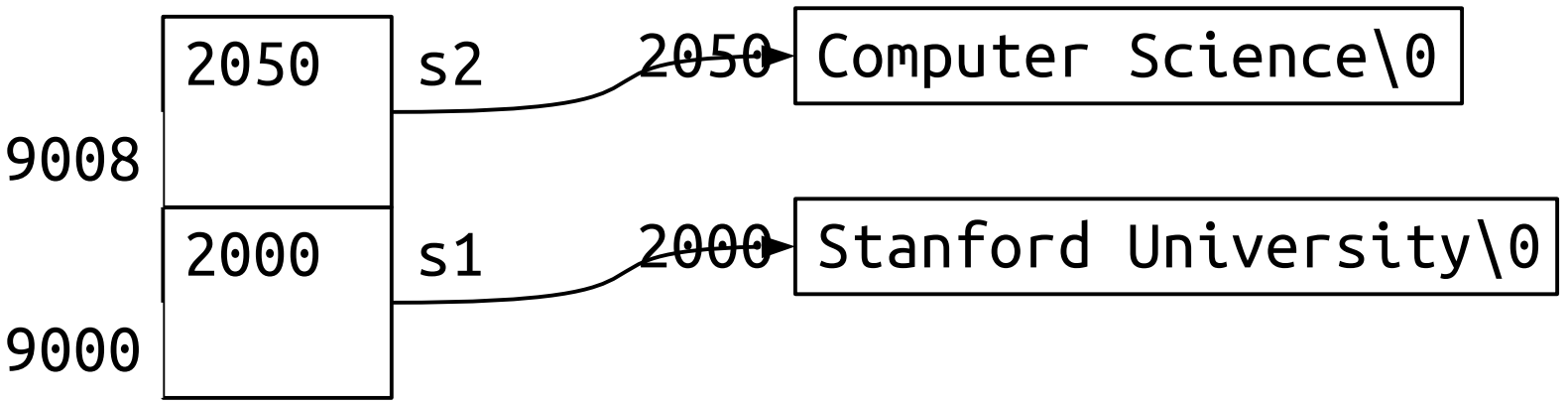



```
char *s1 = strdup("Stanford University");
char *s2 = strdup("Computer Science");


gswap(&s1, &s2, sizeof(char *)); // (1)


free(s1);
free(s2);
```

Using gswap



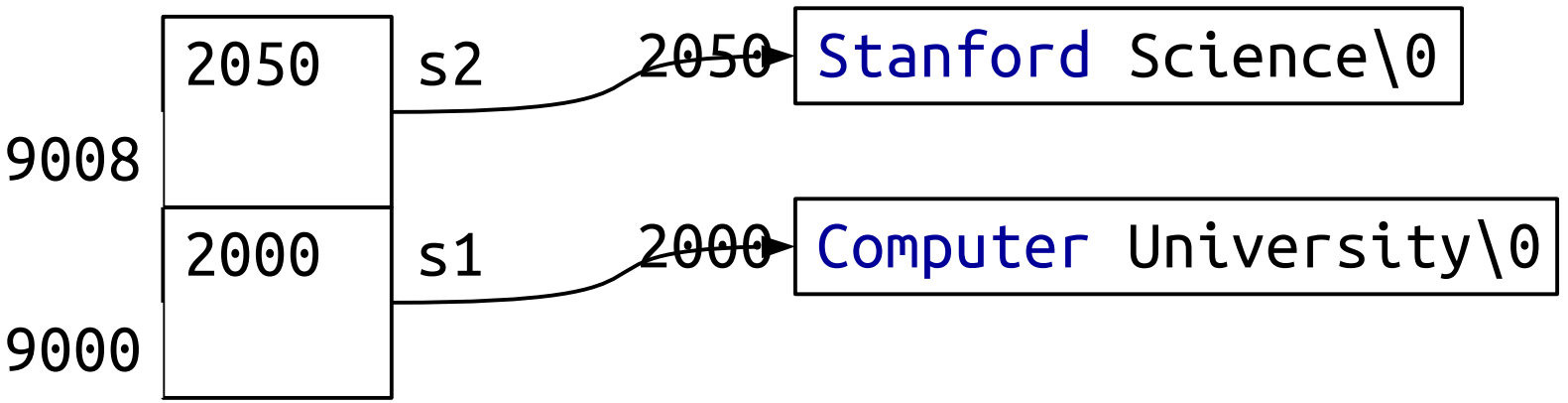

```
char *s1 = strdup(“Stanford University”);
char *s2 = strdup(“Computer Science”);



gswap(s1, s2, sizeof(char *)); // (2)

free(s1);
free(s2);
```

Using gswap




```
char *s1 = strdup("Stanford University");
char *s2 = strdup("Computer Science");



gswap(s1, s2, sizeof(char *)); // (2)

free(s1);
free(s2);
```

Pointers to Elements

Generic functions operate on pointers to elements

If array elements have type	... Pointer to element has type
`int`	`int *`
`double`	`double *`
`char`	`char *`
`char *`	`char **`

Summary

Implement generic functions

- Passing/returning pointers to elements
- Using client-provided callback functions

Discuss limitations of `void *`

- Cannot use pointer arithmetic on `void *`
- Cannot dereference `void *`

Explore level of indirection errors