CIT 5950 Recitation 1

C, Pointers, and Codio

Agenda

1. Logistics
2. Icebreaker
3. Pointer Review
4. C String Review
5. Output parameters Review
6. Codio demo + HW1 Intro

Logistics

Pre Semester Survey
    Due Tuesday January 24th @ 11:59 pm

    (https://canvas.upenn.edu/courses/1702931/quizzes/2982520)

HW0 (Linked List & Hash Table)
    Due Thursday January 26th @ 11:59 pm

Icebreaker!

Break up into groups of ~10

Here are some questions to help you guys get to know each other…

- What's your favorite food
- What would you do with your life if you didn't have to worry about salary?
- If you were sent to a deserted island and could only bring three movies, what would they be?
- Where do you put the '*' in a pointer declaration:
  - `int* ptr,int *ptr,int * ptr,`etc.

Pointer Review

Pointers

Pointers are just another primitive data type.

An integer can hold an index into an array.

If memory is a giant array of bytes, then a pointer just holds an index into that array.

```
type *name;
```
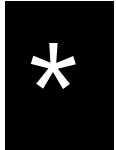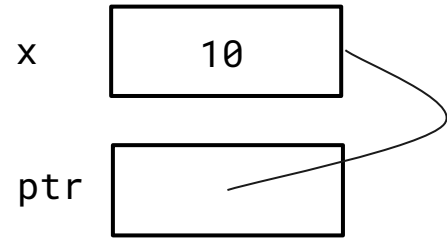
```
int32_t *ptr;
```

ptr | 0x7ff.... |

ptr | ——————→ |

Pointer Syntax

&   "Address of"

*   "Value at"

```
int32_t x;
int32_t *ptr;

ptr = &x;
x = 5;
*ptr = 10;
```
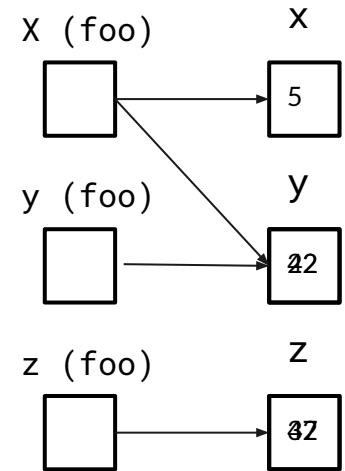
x   [10]

ptr [   ]

Exercise 1

Draw a memory diagram like the one above for the following code
and determine what the output will be.

```c
void foo(int32_t *x, int32_t *y, int32_t *z) {
  x = y;
  *x = *z;
  *z = 37;
}

int main(int argc, char *argv[]) {
  int32_t x = 5, y = 22, z = 42;
  foo(&x, &y, &z);
  printf("%d, %d, %d\n", x, y, z);
  return EXIT_SUCCESS;
}
```

X (foo)   x

                 5

y (foo)   y

                 42

z (foo)   z

                 37

So, the code will output 5, 42, 37.

C-Strings

C-Strings

```
char str_name[size];
```

- A string in C is declared as an array of characters that is terminated by a null character '\0'.
- When allocating space for a string, remember to add an extra character for the null terminator.
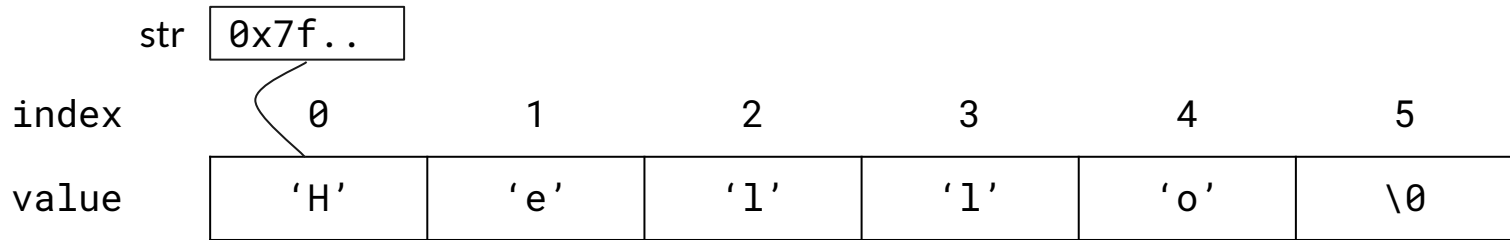
Example

```
char str[6] = "Hello";
```

index	0	1	2	3	4	5
value	'H'	'e'	'l'	'l'	'o'	\0

- If using String literals, C will set it up for you

Example

```
char* str = "Hello";
```

str	0x7f..

index	0	1	2	3	4	5
value	'H'	'e'	'l'	'l'	'o'	\0

- You can also use a pointer. C will allocate the characters in read only memory, and the pointer will point to the first character in the string.

Exercise 1 b

The following code has a bug. What's the problem, and how would you fix it?

```
void bar(char *str) {
  str = "ok bye!";
}

int main(int argc, char *argv[]) {
  char *str = "hello world!";
  bar(str);
  printf("%s\n", str);  // should print "ok bye!"
  return EXIT_SUCCESS;
}
```

main stack frame

bar stack frame

Modifying the argument `str` in `bar` will not effect `str` in `main` because arguments in C are always passed by value.
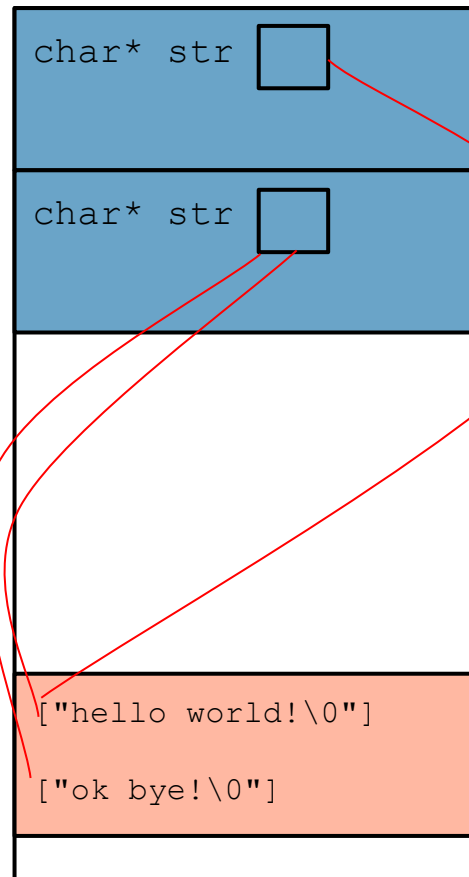
In order to modify `str` in `main`, we need to pass a pointer to a pointer (`char **`) into `bar` and then dereference it:

```
void bar_fixed(char **str_ptr) {
  *str_ptr = "ok bye!";
}
```

static data


```
char* str  [ ]

char* str  [ ]

["hello world!\0"]

["ok bye!\0"]
```

15

Output Parameters

Output Parameters

Definition: a pointer parameter used to store output in a location specified by the caller.

Useful for returning multiple items :)

Output Parameter example

Consider the following function:

```
void getFive(int ret){
    ret = 5;
}
```

Will the user get the value '5'?

No! You need to use a pointer so that the caller can see the change

```
void getFive(int* ret){
    *ret = 5;
}
```

Exercise 2

```
char *strcpy(char *dest, char *src) {
    char *ret_value = dest;
    while (*src != '\0') {
      *dest = *src;
      src++;
      dest++;
    }
    *dest = '\0';  // don't forget the null terminator!
    return ret_value;
}
```

How is the caller able to see the changes in `dest` if C is pass-by-value?

The caller can see the copied over string in `dest` since we are dereferencing `dest`. Note that modifications to dest that do not dereference will not be seen by the caller(such as `dest++`). Also note that if you used array syntax, then `dest[i]` is equivalent to `*(dest+i)`.

Why do we need an output parameter? Why can't we just return an array we create in strcpy?

If we allocate an array inside strcpy, it will be allocated on the stack. Thus, we have no control over this memory after strcpy returns, which means we can't safely use the array whose address we've returned.

Codio Demo

Exercise 3

```
void product_and_sum(int *input, int length, int *product, int *sum) {
    int temp_sum = 0;
    int temp_product = 1;
    for (int i = 0; i < length; i++) {
        temp_sum += input[i];
        temp_product *= input[i];
    }
    *sum = temp_sum;
    *product = temp_product;

}
```