Week #10

Kyle Dewey

Overview

- Meeting times
- Typos
- Project #3 tidbits
- Exam Review

Meeting Times

- Earliest I can possibly meet with anyone is Wednesday after lab
- I will be in my office Thursday (Phelps 1410) from 10 AM - 2 PM
- I can meet from 3 - 7 PM Friday
- ...and that's it

Typos

- Week 9 part 1
- Exam review
  - “struct named bar”
  - Syntax error with pointer to struct
- Look online for latest versions

Project #3

Planning

- Hours of coding can save minutes of planning
- Longer functions = more code in one place = more that needs to be read to understand a piece of code = more headaches
- More variables in one place = more possible interactions = more headaches

Simplicity

- Simpler designs are usually more reliable
- Simplicity is often initially more difficult to achieve than complexity
  - But it pays off in the long run

Functions

```
int main() {
  int input;
  int n = 0;
  int s = 0;
  while ( scanf( "%i", &input ) == 1 ) {
    s += input;
    n++;
  }
  printf( "%i\n", s / n );
  return 0;
}
```

Functions

```
int averageFromInput() {
  int input;
  int n = 0;
  int s = 0;
  while ( scanf( “%i”, &input ) == 1 ) {
    s += input;
    n++;
  }
  return s;
}
int main() {
  printf( “%i\n”, averageFromInput() );
  return 0;
}
```

Functions

```
int main() {
  int arr[] = { 1, 2 };
  int temp = arr[ 0 ];
  arr[ 0 ] = arr[ 1 ];
  arr[ 1 ] = temp;
  temp = arr[ 1 ];
  arr[ 1 ] = arr[ 0 ];
  arr[ 0 ] = temp;
}
```

Functions

```
void swap( int* arr,
           int pos1, int pos2 ) {
  int temp = arr[ pos1 ];
  arr[ pos1 ] = arr[ pos2 ];
  arr[ pos2 ] = temp;
}
int main() {
  int arr[] = { 1, 2 };
  swap( arr, 0, 1 );
  swap( arr, 1, 0 );
}
```

...or don't...

- We aren't grading you on your code structure
- ...but more complexity = more room for bugs
    - Refactoring: rewriting code to have the same functionality but be simpler

Common Pitfalls

“What should this look like?”

- There is too much possible variation in this project to answer this precisely
- I can look at your code to try to understand your approach
  - Different approaches will vary WIDELY in their implementation

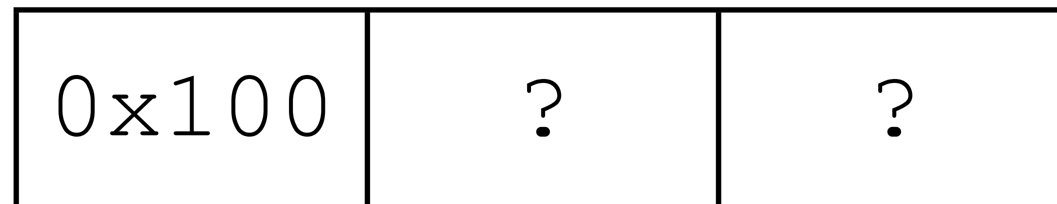
Reusing Memory

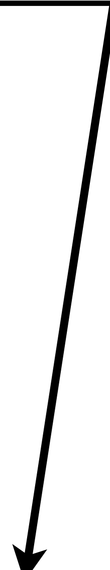
```
char* dictionary[ MAX_NUM_WORDS ];

void readWordsFromTerminal() {
  int x = 0;
  char input[ INPUT_SIZE ];
  while( scanf( “%s”,input ) == 1 ) {
    dictionary[ x ] = input;
    x++;
  }
}
```

First Word

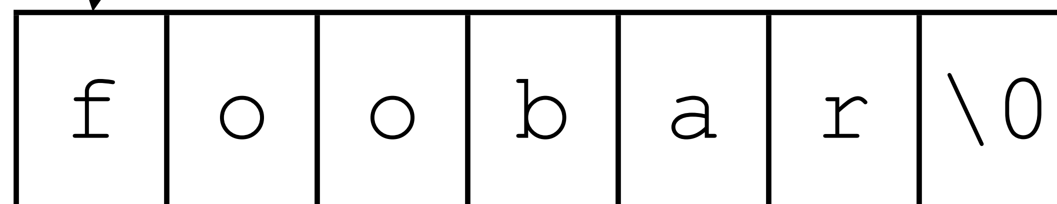
```
scanf( “%s”,input );
dictionary[ 0 ] = input;
```

Dictionary:





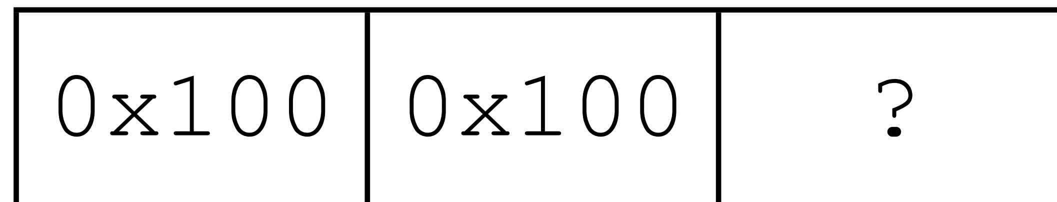
Input:
(at 0x100)

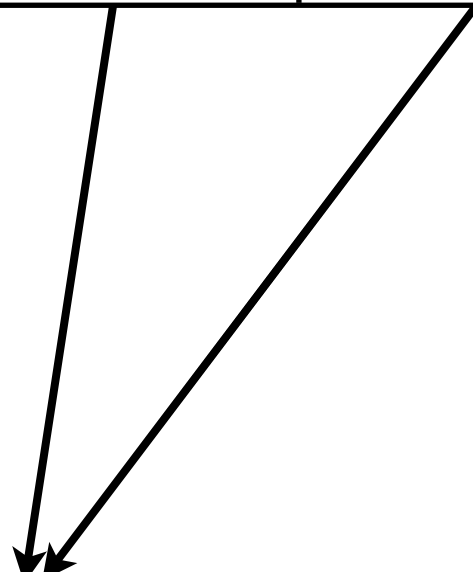


Second Word

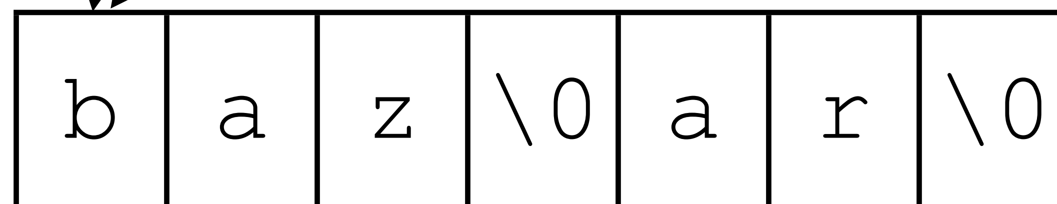
```
scanf( "%s",input );
dictionary[ 1 ] = input;
```

Dictionary:





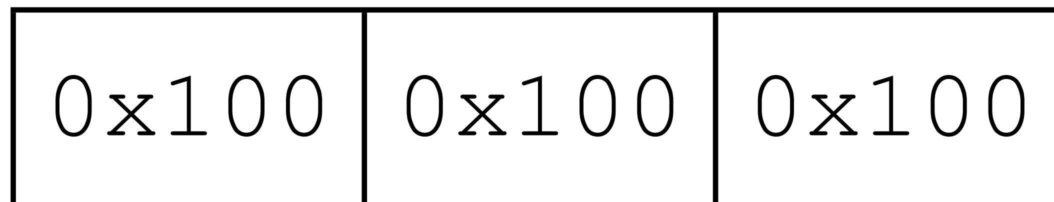
Input:
(at 0x100)

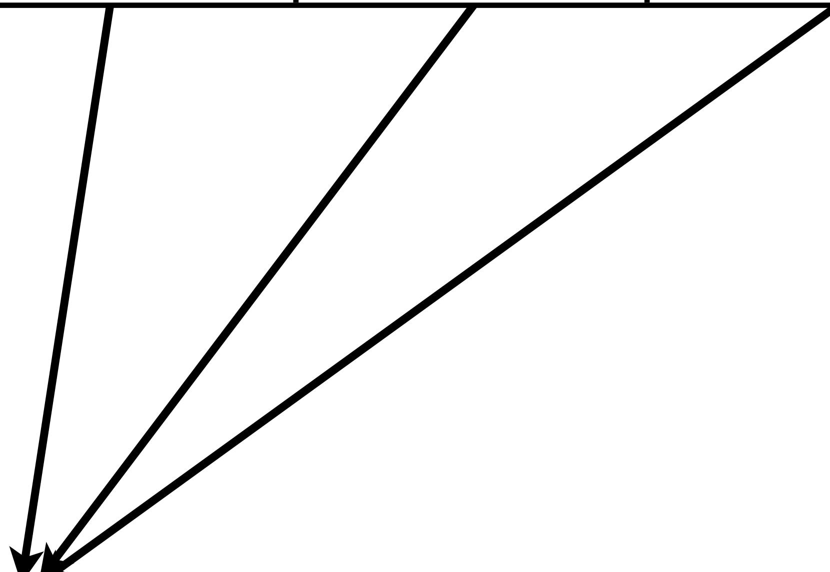


Third Word

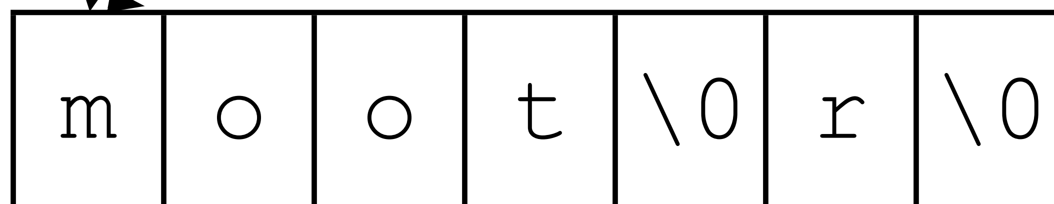
```
scanf( "%s",input );
dictionary[ 2 ] = input;
```

Dictionary:





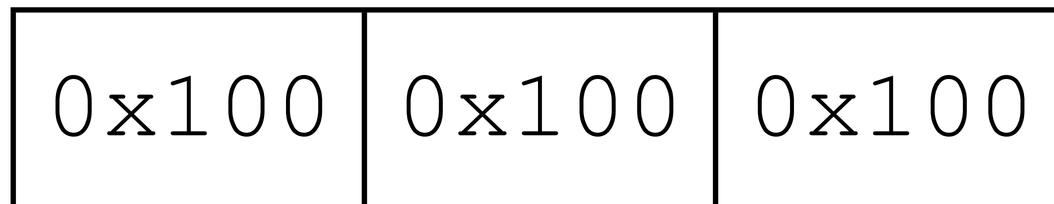
Input:
(at 0x100)



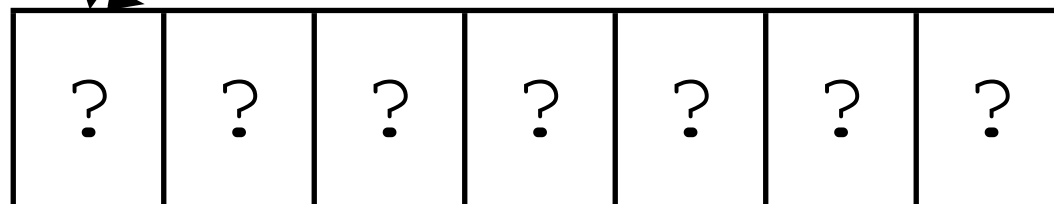
Function Returns

```
{ ... char input[ INPUT_SIZE ]; ... }
```

Dictionary:



Input:
(at 0x100)



(no longer alive)

Comparing Strings

```
char input[ 10 ];
scanf( “%s”, input ); // inputs “foo”
input == “exit” // true or false?
```

Comparing Strings

```
char input[ 10 ];
scanf( “%s”, input ); // inputs “exit”
input == “exit” // true or false?
```

What's Happening

`input:`
(at 0x100)

?	?	?	?	?	?	?	?	?	?

Type: `char[]`

exit:
(at 0x200)

e	x	i	t	\0

Type: `char[]`

What's Happening

`input:`
(at 0x100)

?	?	?	?	?	?	?	?	?	?

Type: `char[]`

exit:
(at 0x200)

e	x	i	t	\0

Type: `char[]`

```
input == "exit"
```

What’s Happening

`input:` (at 0x100)

?	?	?	?	?	?	?	?	?	?

Type: `char[]`

exit: (at 0x200)

e	x	i	t	\0

Type: `char[]`

`input == “exit”`

`0x100 == 0x200?`

What’s Happening

- The addresses themselves are being compared
- The content of the arrays is completely ignored

Comparing Strings

- The correct way: use `strcmp`
- If `strcmp` returns `0`, it means the string **contents** are the same

Comparing Strings

```
char input[ 10 ];
scanf( “%s”, input ); // inputs “foo”
strcmp( input, “exit” ); // 0?
```

Comparing Strings

```
char input[ 10 ];
scanf( “%s”, input ); // inputs “exit”
strcmp( input, “exit” ); // 0?
```

Using `realloc`

- You will almost certainly need to use `realloc`
- Look at `stddev_class_version.c` and `stddev_my_version.c` for more information (specifically the `readNumbers` function)

Strings With Constant Max Length

- You **can** use a declaration like this:

```
char dictionary[ 100 ][ 100 ];
scanf( “%s”, dictionary[ 0 ] );//word 1
scanf( “%s”, dictionary[ 1 ] );//word 2
...
```

- Wastes memory, but that’s ok for this assignment

Exam #3 Review