

What is your experience with
C before this class? (c++ does not count)

clicker.cs.illinois.edu

Q1

~Code~
340



CS 340

C without the ++ (0b01)

me



**What percentage of the class
do you think has successfully
written a program in C?**

clicker.cs.illinois.edu

Q2

~Code~
340



Updates

1. MP 0 is out

Feb 3rd


2. MP 1 is out

3. HW 0 is due Wednesday at midnight

4. Clickers for week 1 uploaded on PL

C without the C++

LGs:

- Be able to read and understand C code
 - Be able to write C code from scratch
1. Why C 
 2. Memory Foundations Review
 3. Structs
 4. Data in C/C++
 5. Dynamic Memory
 6. BST Example

But why C?

lower-level - doesn't have advanced features

C networking, security, embedded systems, OS

↓
pic req

[Review] Running a C/C++ file

1. Compile \rightarrow executable

1. compile each file

\downarrow
v
c

func
declaration
in
each file

gcc \sim .c \sim .c
 \uparrow

2. link together

func
definitions

2 \rightarrow Run executable file

./a.out

[Review] Memory Foundations

```
int x = 3;
```

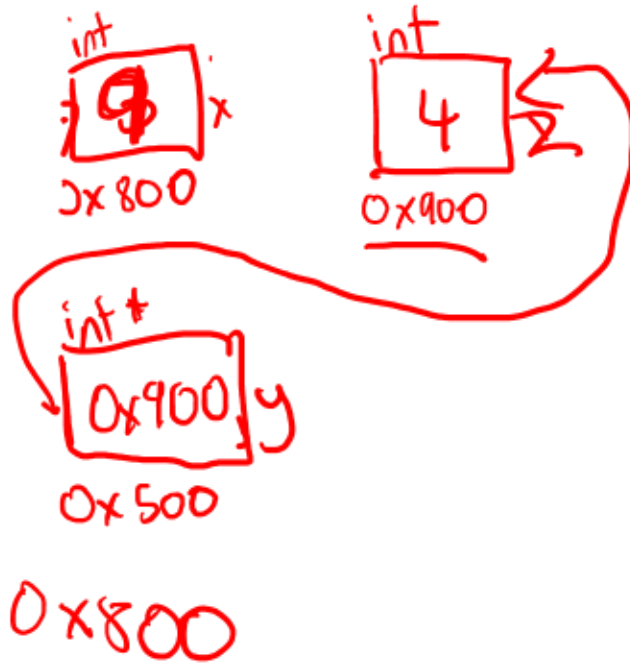
```
int z = 4;
```

```
int *y = &x;
```

```
printf("%x", (y));
```

```
[*y = 9;
```

```
[y = &z;
```



[Review] Memory Foundations w/Functions

```
void foo(int x, int *ptr) {  
    *ptr = 11;  
}
```

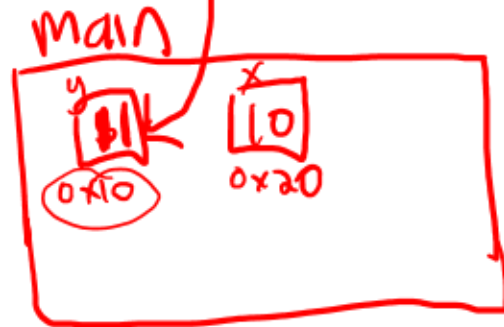


```
int main() {
```

```
    int y = 5;
```

```
    int x = 10;
```

```
    foo(x, y);
```



```
}
```


How is C different from C++?

1. No classes
2. No Templates
3. No function or operator overloading
4. No new or delete
5. No pass-by-reference
6. No standard C++ library → strings
vectors
cin/cout

How confused/nervous are you to code in C without the ++ features?



No classes

Structs

no member functions

Is this valid C code?

```
4 struct food {  
5     int amount;  
6     int age;  
7  
8     int can_eat(){  
9         if(amount > 0 && age < 10){  
10             return 1;  
11         }  
12         return 0;  
13     }  
14 };
```

clicker.cs.illinois.edu

Q3

~Code~
340



NO

Working with Structs in C

Instead of making a class with member functions...

- Create a struct with ONLY member variables.
- Write normal functions that take in an extra parameter.

```
typedef struct food {  
    int amount;  
    int age;  
} Food;  
↑
```

```
int can_eat(food *self){  
    if (self->amount > 0){  
        return 1;  
    }  
    return 0;  
}
```

```
int main() {  
    food test;  
    [test.amount = 0;  
    [can_eat(&test);  
}
```

[Review] -> versus .

$X \rightarrow b \iff (*X).b$
↑
is a
pointer

$X \circ b$
↑
is not
a pointer

What would go in each box?

```
4  typedef struct food {
5      int amount;
6      int *age;
7  } food;
8
9  int main(){
10     int x = 5;
11     struct food fd;
12     fd.amount = 2;
13     fd.age = &x;
14     food *fd_p = &fd;
15     *(fd_p.age) = 7;
16 }
```

Handwritten annotations in red:

- An arrow points from the `age` field in the struct definition to the `age` field in the struct initialization.
- The word `struct` is written next to `food fd;`.
- The `fd` variable is circled in line 12, with an arrow pointing to the `struct food` definition.
- The `fd` variable is circled in line 13, with an arrow pointing to the `struct food` definition.
- The `fd_p` variable is circled in line 14, with an arrow pointing to the `struct food` definition.
- The `age` field in the pointer access is circled in line 15, with an arrow pointing to the `age` field in the struct definition.

clicker.cs.illinois.edu

Q4

~Code~
340



Q5

~Code~
340



```
4  typedef struct food {  
5      int amount;  
6      int age;  
7  } food;  
8  
9  void food_init(food *self, int am){  
10     self->amount = am;  
11     self->age = 0;  
12 }
```

Handwritten annotations: Red arrows point from the underlined `food_init` to the `self` parameter and the `am` parameter. A red circle labeled 'C' is next to the function definition.

```
int main(){  
    food fd = food_init(&fd, 2);  
}
```

Handwritten annotations: Red arrows point from the underlined `food_init` to the `&fd` argument and the `fd` variable. A red arrow points from the underlined `food fd` to the `fd` argument.

(A)

```
int main(){  
    food fd;  
    food_init(&fd, 2);  
}
```

Handwritten annotations: A red circle labeled 'A' is around the entire code block. A red arrow points from the underlined `food fd` to the `fd` argument in `food_init`.

(B)

```
int main(){  
    food fd(2);  
}
```

Handwritten annotations: A red circle labeled 'B' is next to the code block. A red 'X' is to the right of the code, indicating it is incorrect.

Data in C/C++

Data Stored as Bytes

Physically, computer hardware...

high, low, 1, 0

↑
store things

To make binary more practical we...

8 bits \rightarrow 1 byte

Data types have a size (in bytes)...

char = 1 byte = 8 bits

int = 4 byte = 32 bits

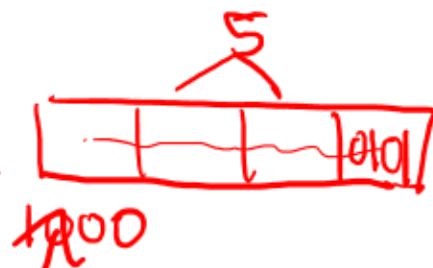
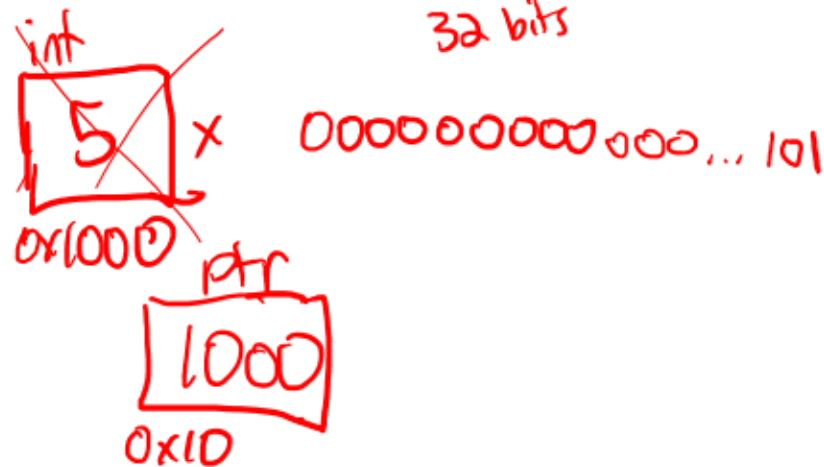
Data Stored as Bytes

`int x = 5`

`int *ptr = &x;`

An address is a number

Bytes 10-18



If a pointer is 8 bytes, how many bits is it?

64

00101010...

clicker.cs.illinois.edu

Q6

~Code~
340



Dynamic Memory

[Review] What is Dynamic Memory

request memory at run time

- flexible to change memory used while the code is running
- control objects life time

//allocates size bytes on the heap and returns a
//pointer to that memory location on the heap.

void *malloc(size_t size);

//frees the memory at ptr from the heap
void free(void *ptr);

sizeof(int)
↓
int *ptr = malloc(4);
free(ptr);

What is wrong?

```
1  #include <stdlib.h>
2
3  int main(){
4      int x = 5;
5      → int *ptr = malloc(sizeof(int));
6      → ptr = &x;
7      → free(ptr);
8  }
```

Memory error

clicker.cs.illinois.edu

Q7

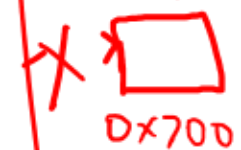
~Code~
340



stack



heap



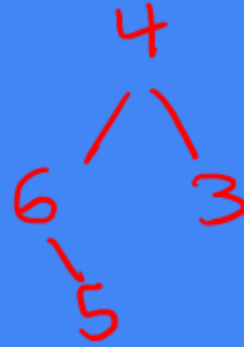
[Review] What are some common memory management errors?

Finding Memory Errors

valgrind --leak-check=full --show-leak-kinds=all ./exec

shell

```
==10389== HEAP SUMMARY:
==10389==      in use at exit: 24 bytes in 1 blocks
==10389==    total heap usage: 1 allocs, 0 frees, 24 bytes allocated
==10389==
==10389== 24 bytes in 1 blocks are definitely lost in loss record 1 of 1
==10389==    at 0x488545C: malloc (vg_replace_malloc.c:446)
==10389==    by 0x4006C3: add_node_helper (bst.c:21)
==10389==    by 0x40076B: add_node (bst.c:37)
==10389==    by 0x40080B: main (bst.c:58)
==10389==
==10389== LEAK SUMMARY:
==10389==    definitely lost: 24 bytes in 1 blocks
==10389==    indirectly lost: 0 bytes in 0 blocks
==10389==    possibly lost: 0 bytes in 0 blocks
==10389==    still reachable: 0 bytes in 0 blocks
==10389==    suppressed: 0 bytes in 0 blocks
==10389==
==10389== For lists of detected and suppressed errors, rerun with: -s
==10389== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
```



BST Example

What goes in the box?

```
5  typedef struct node {  
6      struct node *left;  
7      struct node *right;  
8      int datum;  
9  } node;  
10  
11  typedef struct bst {  
12      struct node *root;  
13  } bst;  
14  
15  void init_bst(bst *self){  
16      [REDACTED]  
17  }
```

clicker.cs.illinois.edu

Q8

~Code~
340



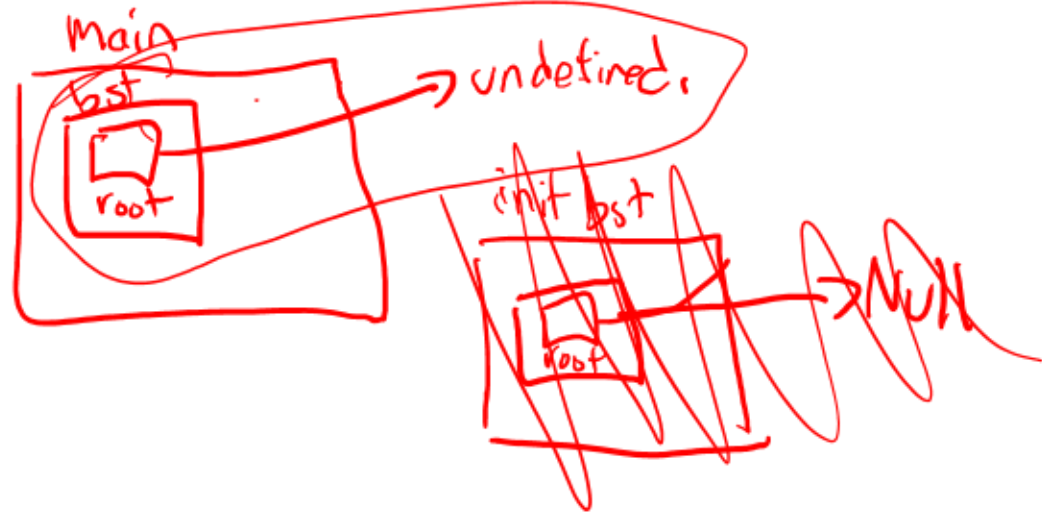
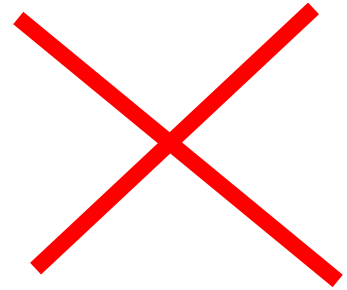
self->root = NULL;

```
5 typedef struct node {
6     struct node *left;
7     struct node *right;
8     int datum;
9 } node;
```

```
11 typedef struct bst {
12     struct node *root;
13 } bst;
```

```
15 void init_bst(bst self) {
16     self.root = NULL;
17 }
```

```
int main() {
    bst b;
    init_bst(b);
    return 0;
}
```



```
5  typedef struct node {  
6      struct node *left;  
7      struct node *right;  
8      int datum;  
9  } node;
```

```
10  
11  typedef struct bst {  
12      struct node *root;  
13  } bst;
```

```
14  
15  void init_bst(bst *self){  
16      self->root = NULL;  
17  }
```

```
int main() {  
    bst b;  
    init_bst(&b);  
    return 0;  
}
```

What goes in the box?

clicker.cs.illinois.edu

Q9

~Code~
340



```
void add_node(bst *self, int data){  
    self->root = add_node_helper(self->root, data);  
}
```

```
19 node *add_node_helper(node *node_ptr, int data){  
20     if(node_ptr == NULL){  
21         node *tmp = malloc( XXXXXXXXXX );  
22         tmp->left = NULL;  
23         tmp->right = NULL;  
24         tmp->datum = data;  
25         return tmp;  
26     }
```

↑
sizeof(node)

**Given the code so far, does
bst have any member
functions?**

clicker.cs.illinois.edu

Q10

~Code~
340



Is the default constructor called?

```
40  ✓ int main() {  
41      bst b;  
42      add_node(&b, 4);  
43      add_node(&b, 2);  
44      add_node(&b, 3);  
45      return 0;  
46  }
```

clicker.cs.illinois.edu

Q11

~Code~
340



```
typedef struct bst {  
    struct node *root;  
} bst;
```

Is a destructor called for bst?

```
40  ✓ int main() {  
41      bst b;  
42      add_node(&b, 4);  
43      add_node(&b, 2);  
44      add_node(&b, 3);  
45      return 0;  
46  }
```

clicker.cs.illinois.edu

Q12

~Code~
340



```
typedef struct bst {  
    struct node *root;  
} bst;
```

```
40 void bst_pop_all() {  
41       
42       
43       
44       
45       
46       
47       
48 }  
49  
50 void bst_destruct() {  
51       
52 }
```

clicker.cs.illinois.edu

Q13

~Code~
340



```
typedef struct bst {  
    struct node *root;  
} bst;
```

```
typedef struct node {  
    struct node *left;  
    struct node *right;  
    int datum;  
} node;
```

Finding Memory Errors

`valgrind --leak-check=full --show-leak-kinds=all ./exec`

```
==10389== HEAP SUMMARY:
==10389==      in use at exit: 24 bytes in 1 blocks
==10389==    total heap usage: 1 allocs, 0 frees, 24 bytes allocated
==10389==
==10389== 24 bytes in 1 blocks are definitely lost in loss record 1 of 1
==10389==    at 0x488545C: malloc (vg_replace_malloc.c:446)
==10389==    by 0x4006C3: add_node_helper (bst.c:21)
==10389==    by 0x40076B: add_node (bst.c:37)
==10389==    by 0x40080B: main (bst.c:58)
==10389==
==10389== LEAK SUMMARY:
==10389==    definitely lost: 24 bytes in 1 blocks
==10389==    indirectly lost: 0 bytes in 0 blocks
==10389==    possibly lost: 0 bytes in 0 blocks
==10389==    still reachable: 0 bytes in 0 blocks
==10389==    suppressed: 0 bytes in 0 blocks
==10389==
==10389== For lists of detected and suppressed errors, rerun with: -s
==10389== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
```

C without the C++

LGs:

- Be able to read and understand C code
- Be able to write C code from scratch

1. Why C
2. Memory Foundations Review
3. Structs
4. Data in C/C++
5. Dynamic Memory
6. BST Example