

The end of CS 126

PICK UP ICES

What was CS 126 about?

- My goals for this class:

1. **Improve your programming productivity by $\geq 3x$**
2. Build your self-sufficiency as a programmer
3. Introduce you to modern computing environments
4. Provide skills for getting internships / doing hack-a-thons
5. Have you build a large project relating to your interests

6. expose you to C++ syntax.

C++: what does this allocate

IceCreamSandwich myIceCreamSandwich;

- A) A pointer to an ice cream sandwich object on the heap
- B) A pointer to an ice cream sandwich object on the stack
- C) An ice cream sandwich object on the heap
- D) An ice cream sandwich object on the stack
- E) A reference to an ice cream sandwich object on the heap
- F) A reference to an ice cream sandwich object on the stack

Pointers to objects

■ Which of the following allocates a pointer to an object?

A) `IceCreamSandwich myIceCreamSandwich;`

B) `IceCreamSandwich *myIceCreamSandwich;`

C) `IceCreamSandwich &myIceCreamSandwich;`

D) `IceCreamSandwich myIceCreamSandwich*;`

E) `IceCreamSandwich myIceCreamSandwich&;`

Allocating a Fish

Which is the right syntax for allocating a Fish?

A) Fish fish = new Fish();

B) Fish *fish = new Fish(); ← new returns a pointer

C) Fish &fish = new Fish();

D) None of the above

Feeding the dog

Dog fido("Fido"); *object*

If the feed function takes 1 argument, which is a pointer to a dog, how do I call it?

A) feed(fido);

B) feed(*fido);

C) feed(&fido);

D) feed.fido;

E) feed->fido;

dereference (pointer → object)

address of operator (object → pointer)

Feeding the dog, continued.

Dog fido("Fido");

non-static

If the feed function is a method of Dog, how do I call it?
n

- A) `fido->feed();` // pointer *Dog * fido;*
- B) `fido.feed();` // object
- C) `(*fido).feed();` // pointer *Dog * fido;*
- D) `(*fido)->feed();` // pointer to a pointer *Dog ** fido;*
- E) None of the above

Setting values

Which of the following statements should be used in this function?

```
void exampleFunction(int *thing) {
```

A) `thing = 7;`

B) `thing = *7;`

C) `thing = &7;`

D) `&thing = 7;`

E) `*thing = 7;`

Destroying Objects

```
FooBar *foo = new FooBar();  
// how do we destroy foo?
```

A) delete foo;

B) foo->delete();

C) ~foo;

D) ~foo();

E) foo->~FooBar();

& functions

By default, member variables of a class are:

in C++

- A) public**
- B) private**
- C) protected**
- D) It is undefined**

The << operator

What does the << operator do?

- A) Reads from an input stream
- B) Writes to an output stream
- C) It depends

00100000 =
8 ↗ 32 ↘
8 << 2 = 32

`cout << 7 << endl;`

In C++, what will the following code print out?

```
void  
main() {  
    int x;  
    std::cout << x << std::endl;  
}
```

A) 0

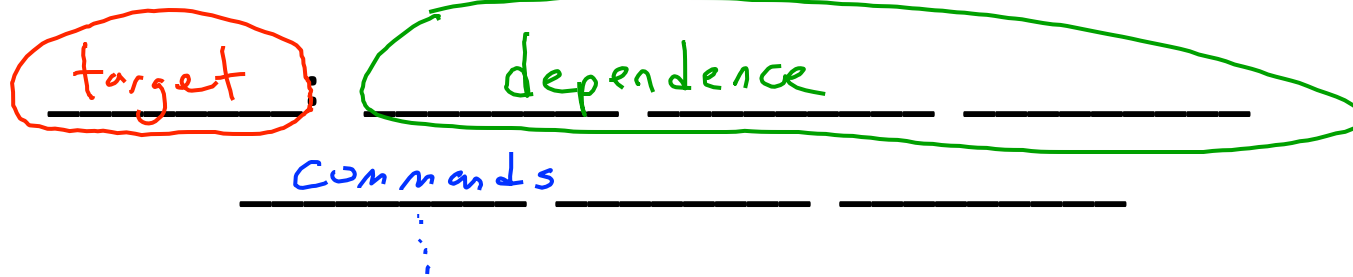
B) NULL

The first 3 are actual values

C) undefined

D) *The value printed is undefined*

Makefiles



A) Dependences

B) Executable

C) Target

D) Commands

E) Directive

```
all : foo bar
```

```
foo :
```

```
bar :
```

Makefiles

```
___A___: ___B___ _____  
      ___C___ _____
```

Where would you put header files?

D) None of the above

```
foo: foo.cpp foo.h  
    clang -o foo foo.cpp
```

What are these?

```
char *str[10];
```

```
int (*q)(int);
```

Clockwise/Spiral Rule

(x+7) y

- <http://c-faq.com/decl/spiral.anderson.html> ←

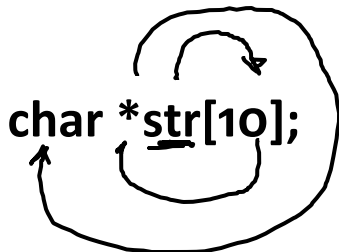
Parse any C declaration in your head!

Starting with the unknown element, move in a spiral/clockwise direction; when encountering the following elements replace them with the corresponding English statements:

1. [X] or [] => Array X size of... or Array undefined size of...
2. (type1, type2) => function passing type1 and type2 returning...
3. * => pointer(s) to...

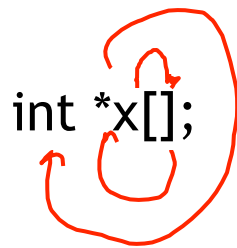
Keep doing this in a spiral/clockwise direction until all tokens have been covered.

Always resolve anything in parenthesis first!

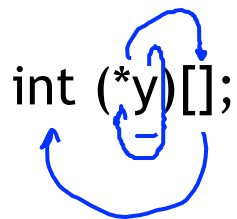


str is an array of pointers to characters.

More Examples (Arrays and Pointers)

`int *x[];`


x is an array of pointers to integers

`int (*y)[];`


y is a pointer to an array of integers.

More Examples (Const and Pointers)

`const char *chptr;`

chptr is a pointer to a character constant

`*chptr = 7;` // not allowed

`const char foo = 7;`
`chptr = &foo;`

`char * const chptr;`

chptr is a constant pointer to character.

`*chptr = 7;` // ok

`chptr = &foo;` // not allowed

More Examples (Functions and Pointers)

```
int *z(int);
```

z is function that takes an integer argument and returns a pointer to an integer.

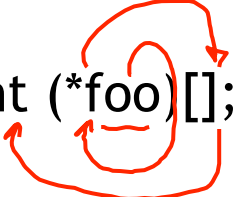
```
int (*q)(int);
```

q is a pointer to a function that takes an integer argument and returns an integer.

~~What does the pointer point to?~~

is foo?

int (*foo)[];



- A) A pointer to an array of integers
- B) An array of pointers to integers
- C) A pointer to a function that takes no arguments and returns an integer
- D) A function that takes no arguments and returns a pointer to an integer
- E) None of the above

Where to go from here?

- **Do some programming over break! (for fun!)**
 - Build an App, learn Javascript, etc.
 - The more you do it, the easier it gets
 - Start building a “portfolio”, have a github presence
- **Try to figure out what part of CS you are interested in**
 - Theory? Systems? Data/ML? HCI? Graphics/VR? Bio?
 - Informational interviews