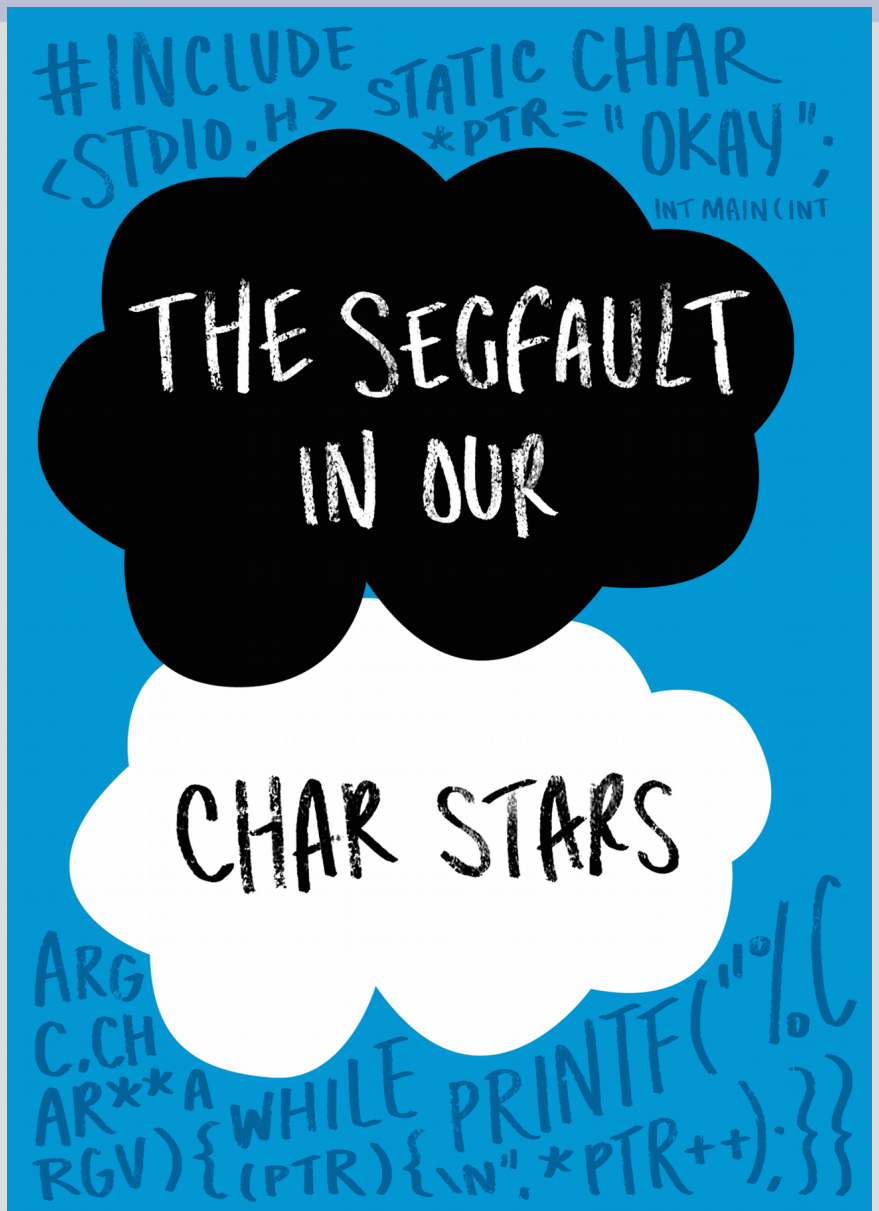
Pointers and memory

Ch 9 & 13.1



Highlights

- new & delete

```
int *xp;
xp = new int;
*xp = 5;
delete xp;
```

Pointers

A pointer is used to store a memory address and denoted by a * (star!)

```
int x=6;
int *xp;
xp = &x;
```

Here variable xp is a integer pointer

```
cout << *(&x); // *(&x) same as  x
```

The * goes from address to variable (much like when you hit ENTER on a url)
(See last time: pointerBasics.cpp)

Person class

How would you make your grandmother? How could you get your grandmother using only yourself as a named object?

```
class person{
      string name;
      person* mother;
      person* father;
};
```

(See: personV2.cpp)

->

When dealing with classes, often you need to deference (*) and access a member (.)

There is a shortcut to de-reference and call a member (follow arrow and go inside a box)

You can replace (*var).x with var->x, so...

```
(*(me.mother)).name;
```

... same as ...

```
me.mother->name;
```

Boxes

What is comes next in this pattern?

Basic programming: `int x;`
Ask for one box with a name

Intermediate programming: `int x[20];`
Ask for multiple boxes with one name

Advanced programming: ???
???

Boxes

What is comes next in this pattern?

Basic programming: `int x;`
Ask for one box with a name

Intermediate programming: `int x[20];`
Ask for multiple boxes with one name

Advanced programming: `new int;`
Ask for a box without giving it a name

new

Pointers are also especially useful to use with the new command

The new command will create a variable (box) of the type you want

```
int x;
x = 2;
```

ask for box

```
int *xp;
xp = new int;
*xp = 4;
```

The new integer has no separate name, just part of xp (as array boxes part of array name) (See: newMemory.cpp)

new

What does this do?

```
int main()
{
    while(true)
    {
        int *x = new int;
    }
    return 0; //totally going to get here!
}
```

new

What does this do?

```
int main()
{
    while(true)
    {
        int *x = new int;
    }
    return 0; //totally going to get here!
}
```

Asking for a lot of boxes there...
(See: memoryLeak.cpp)

delete

When your program exits, the operating system will clean up your memory

If you want to clean up your memory while the program is running, use delete command

```
int *imaPointer; // pointer box (holds address)
imaPointer = new int; // point here!
// do some stuff...
delete imaPointer; // goodbye pointer
```

(See: deleteMemory.cpp)

delete

This is also a memory leak:

```
int *ptr; // make a pointer
ptr = new int; // point here
ptr = new int; // more the merrier
delete ptr; // ERASE
```

By the 3rd line, there is no link back to the box on the 2nd line (dangling pointer)

There should be a “delete” for every “new”

delete

As you can manage how you want to create new variables/boxes, using new/delete is called dynamic memory

Before, the computer took care of memory by creating variables/boxes when you use a type then deleting when the function ends

← Before

Now →

delete

Memory management is a hard part of C++

You need to ensure you delete all your boxes after you are done with them, but before the pointer falls out of scope
(see: lostPointer.cpp)

Some other languages manage memory for you

Person class

The ability to have non-named boxes allows you to more easily initialize pointers

```
class person{
    string name;
    person* mother;
    person* father;
};
```

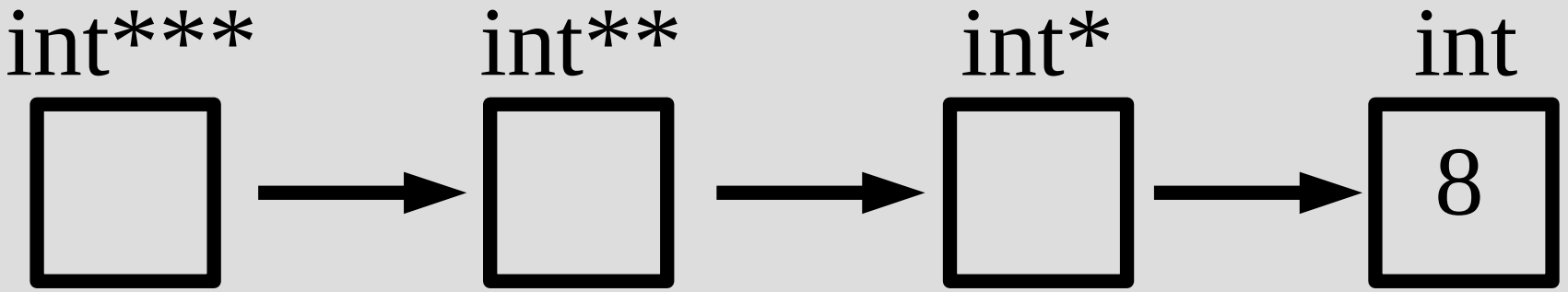
(See: personV3.cpp)

Pointer to pointer

You can have multiple stars next to types:

```
int*** x;
```

Each star indicates **how many arrows** you need to follow before you find the variable



(See: pointerPointers.cpp)