Today - Lecture 12 - CS162

1) Pointer Arithmetic
2) Introduction to "Linear Linked Lists"
3) Demonstrations

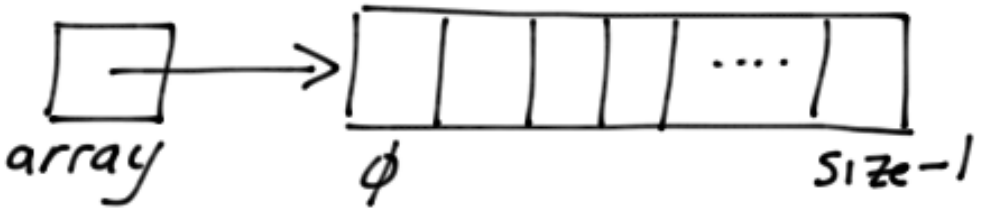
Announcements:

* Remember to spend time with the quiz topics to prepare for the final exam
* Exams are being returned

Dynamically Allocated Arrays

```
char * array = new char[some_size];
```

desired size + 1



Accessing the array can be done through the subscript operator:

```
cout << array[i];
```

displays the character at index i

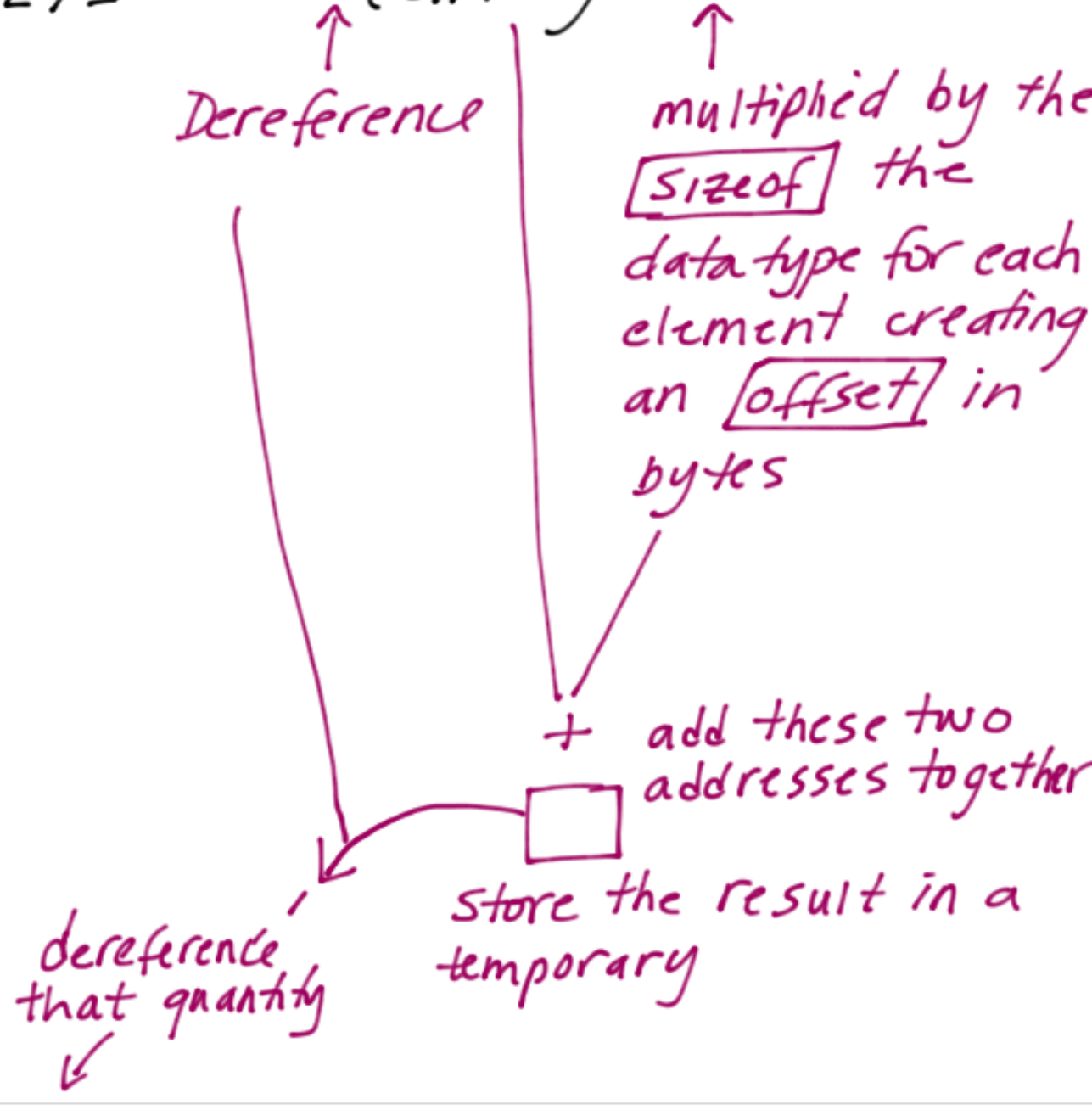
Or, use the cstring library (for arrays of characters):

```
length = strlen(array);
if (strcmp(array, "Karla") == 0)
```

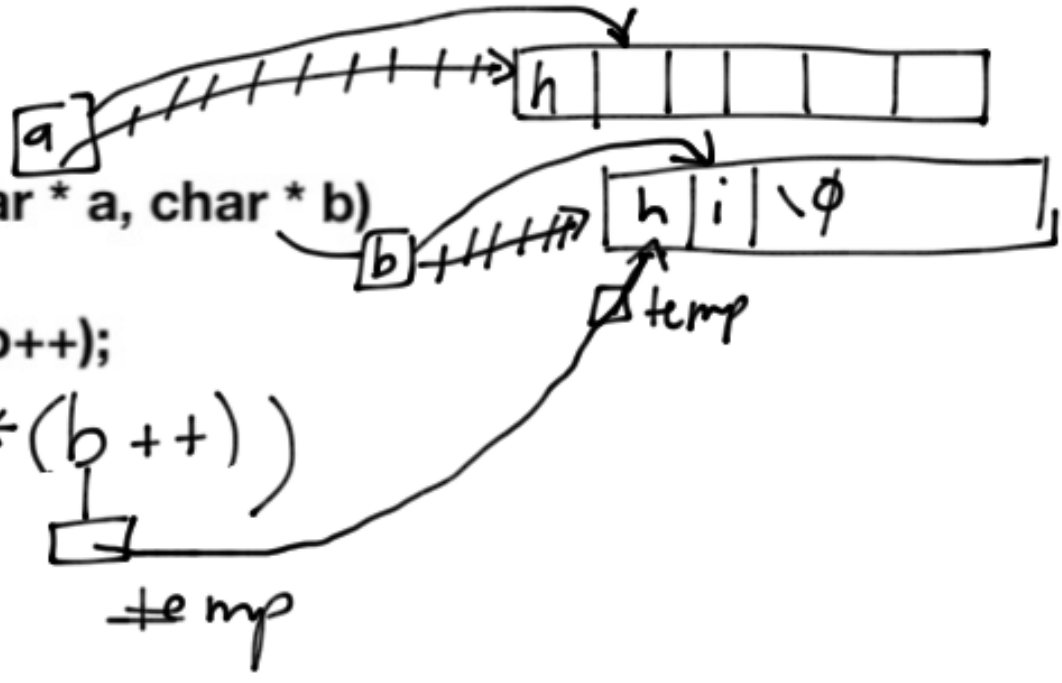
Pointer Arithmetic

The subscript operator actually performs the following actions:

```
array[i] == *(array + i)
```



Look at this code:




```
void function(char * a, char * b)
{
   while(*a++ = *b++);
}
```

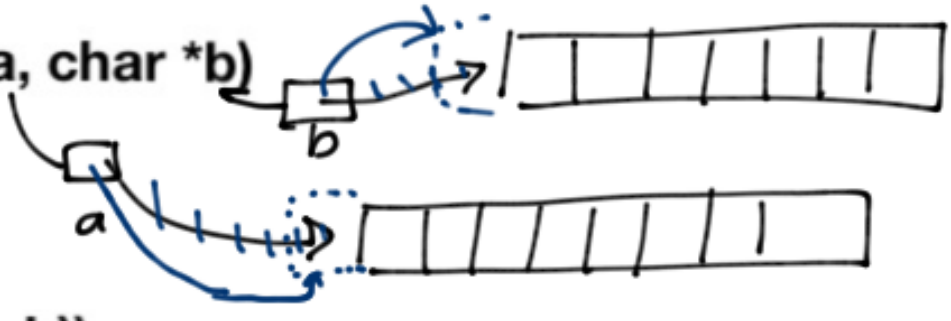
*(a++)

(*(b++))


```
void function (char *a, char *b)
{
    --a; --b;

    while(*(++a)  = *(++b));

}
```



Pointer Arithmetic

```
char name[11];
```

constant pointer to the first element

```
char *ptr = name;
```

Same as:

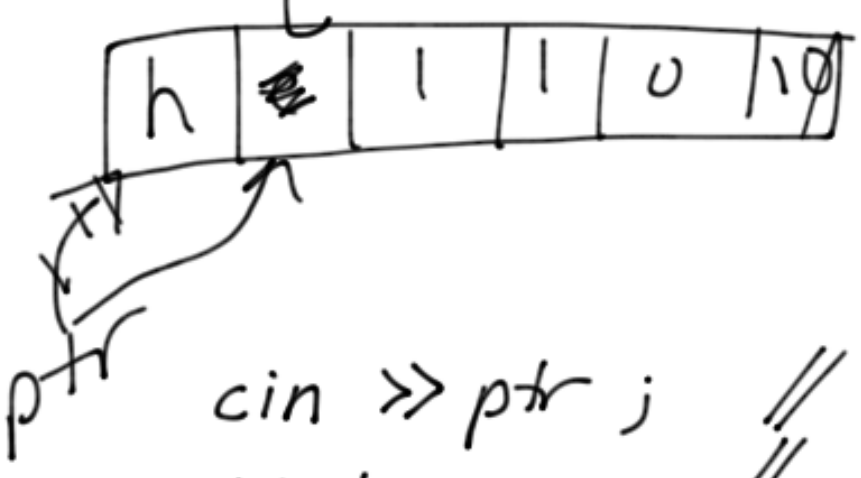
```
ptr = &name[0];
```

& * (name+0)

name

cancel out

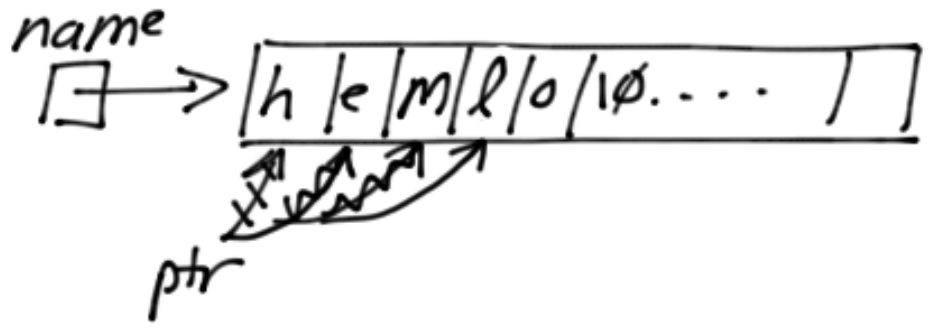
ptr = name




```
cin >> ptr;
++ptr;
*ptr = 'c';
cout << *ptr;
```

```
// cin >> name;
// cout << name;
cout << ptr;
```

Others:




```
char *ptr = name;

cout << ptr;          // hello
cout << *ptr;         // h
cout << *(ptr++);     // h
cout << *(++ptr);     // l
cout << ++(*ptr);     // m
cout << ++ptr;        // lo
```

What is wrong with this:

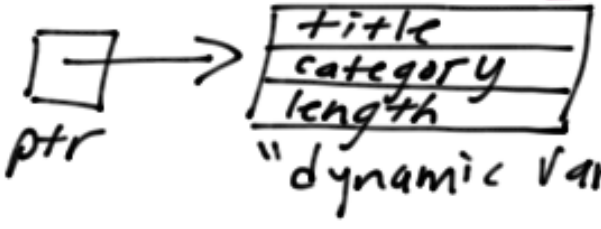
```
*ptr++;          // Same ++ptr;
```

OR

```
*(++ptr);        // same as ++ptr;
```

Pointers and Structures

```
video *ptr = new video;
```



Assume:
char title[131];

```
cout << "Please enter the title";
cin.get(_____, 131);
```

what goes here?

(a) *ptr.title ← doesn't compile

(b) (*ptr).title ← compiles but.....

(c) ptr->title

called the indirect member access operator!

Pointer -> member vs. object . member

Very Important

object . member

struct or class

vs.

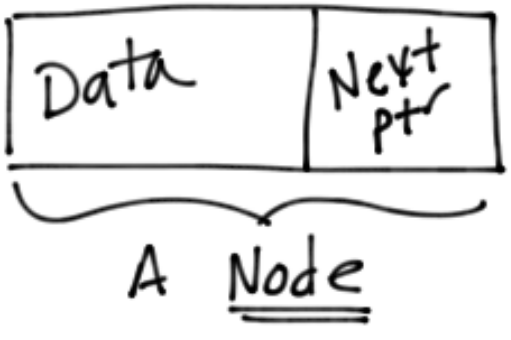
pointer → member

pointer to a struct or class

Make sure the pointer is NOT NULL before dereferencing

Next topic: Linear Linked Lists

1) Flexible
2) Start with nothing & grow/shrink as needed

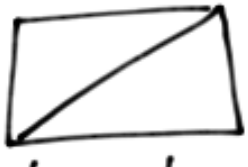



```
struct node
{
        video  show;
        node  * next;   <-- called a recursive definition
};
```

Begins with ...

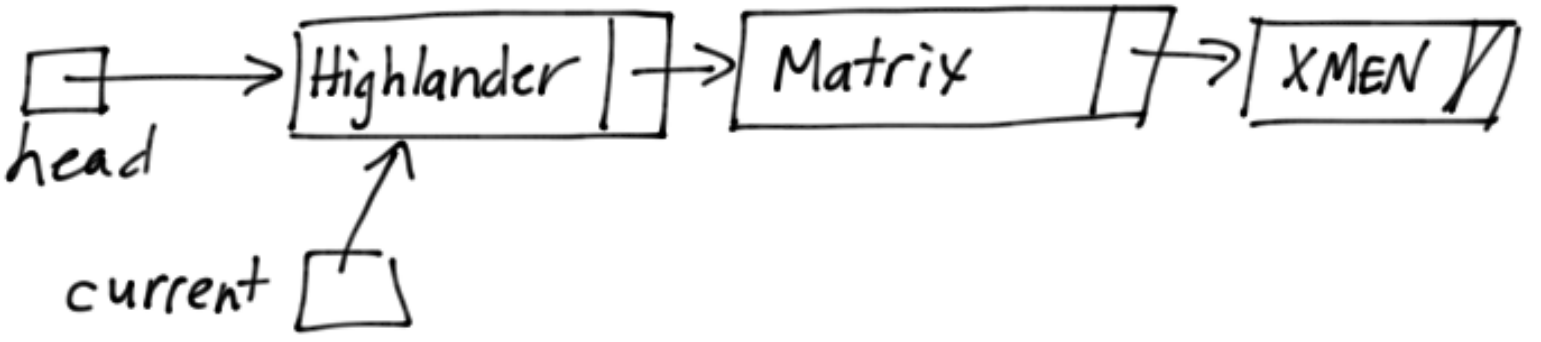
- "head" pointer
- which is a pointer to a node
- initialized to NULL for an empty list
- we can use other pointers to assist with traversal, creation, removal, retrieval

```
node * head = NULL;
```



← represents an empty list. NO Items!

Examine this code to Traverse




```
node * current = head;
while (current != NULL)   //while (current)
{
    cout <<current->show.title <<endl;
    current = current->next;
}
```