Monday, January 29, 2018 9:16 AM

Day's Goals

0) Pointers to user defined types (structs)

1) Dynamically Allocated Memory
   ↳ Array Doubling e.g.

2) Working toward linked lists
   ↳ the Node

Pointers

We have learned that a pointer is described by what type it points.

e.g. int *ptr;

data_type * ptr_name;

int, float, char, bool, void

fundamental data types
  ↳ i.e. basic

or

user defined types
(structs, classes, typedef)

e.g. defined struct:

```
struct Amigo
{ string name;
  int age;
};

int main ( ){
    Amigo jose; // instance of Amigo
    Amigo *josePtr; // pointer for Amigo type
    josePtr = &jose; // assign the reference

    // syntax for accessing members of
// struct via pointer
(*josePtr).name = "Maurinho";
(*josePtr).age = 55;
```

equivalent:

```
josePtr->name = "Maurinho";
josePtr->age = 55;
```

---

e.g.

```
struct ListNode
{
    string userName;
    ListNode *link;
}
```

"circular definition"

<u>Dynamically Allocated</u>

Automatic variables get allocated on the stack.

Dynamically allocated variables get allocated at run-time and use the heap (aka freestore).

"new" and "delete" operators

e.g.

```
int *p1;
p1 = new int;    // nameless variables
```

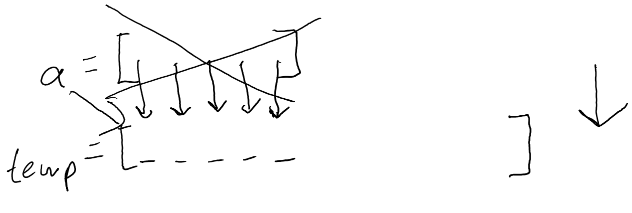
```
delete p1;
~~*p1 = 5;~~   //BAD
```

array e.g. - dynamic array with user-input length

```
int n;
int *ptr;

cin >> n;

ptr = new int[n];

delete [] ptr;
```

e.g. - Create dynamic array length n
- fill w/ user inputs
- Double length of the array
- keep first n values unchanged

```
   int n = 5;
-> int *a;
-> a = new int[n];    // dynamic array length n
   for (i=0, i<n, i++)
       cin >> a[i];

-> int *temp;
   temp = new int[2*n];

   for(i=0, i<n, i++)
       temp[i] = a[i];   <-
   delete [] a;
```



```
delete [] a;
a = temp;
~~delete [] temp;~~ // Don't free this space

temp = nullptr; //NULL
```