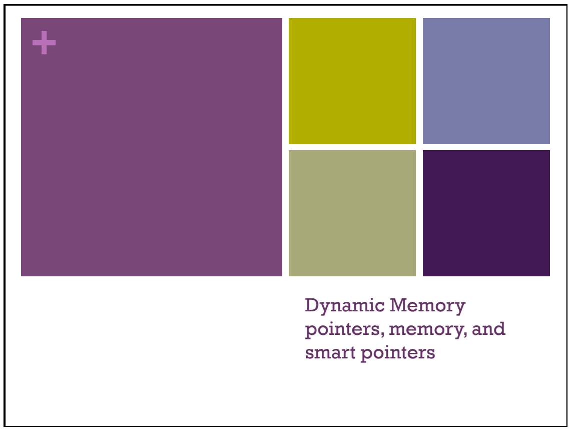
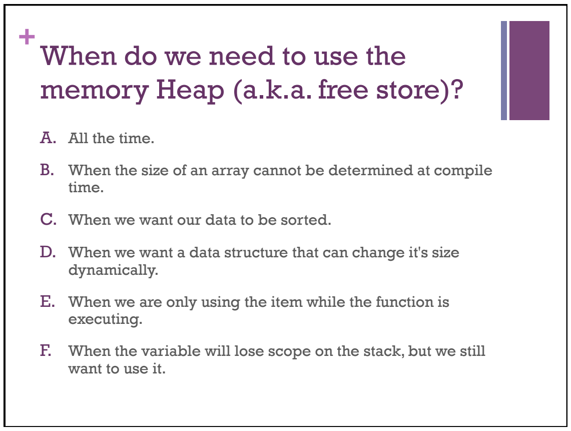


+

Dynamic Memory
pointers, memory, and
smart pointers



+ When do we need to use the memory Heap (a.k.a. free store)?

A. All the time.

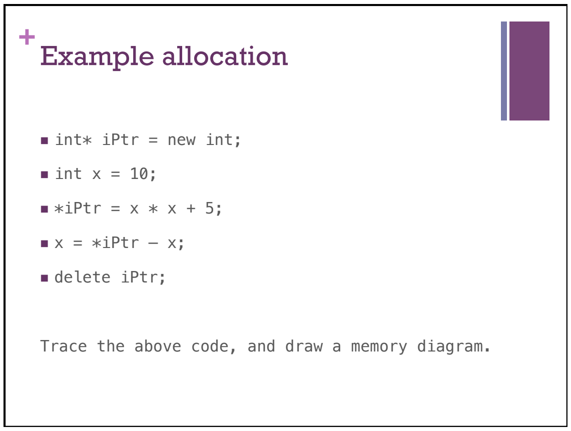
B. When the size of an array cannot be determined at compile time.

C. When we want our data to be sorted.

D. When we want a data structure that can change it's size dynamically.

E. When we are only using the item while the function is executing.

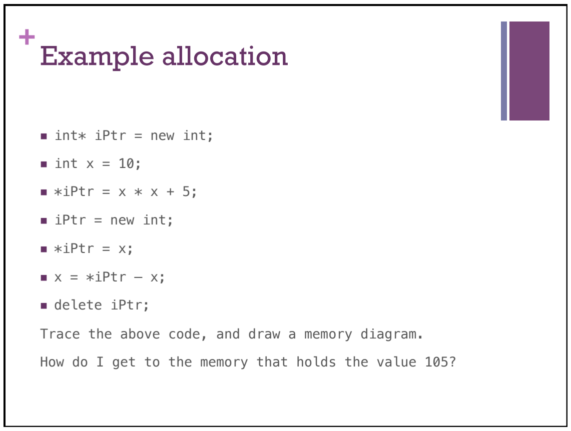
F. When the variable will lose scope on the stack, but we still want to use it.



+ Example allocation

- `int* iPtr = new int;`
- `int x = 10;`
- `*iPtr = x * x + 5;`
- `x = *iPtr − x;`
- `delete iPtr;`

Trace the above code, and draw a memory diagram.

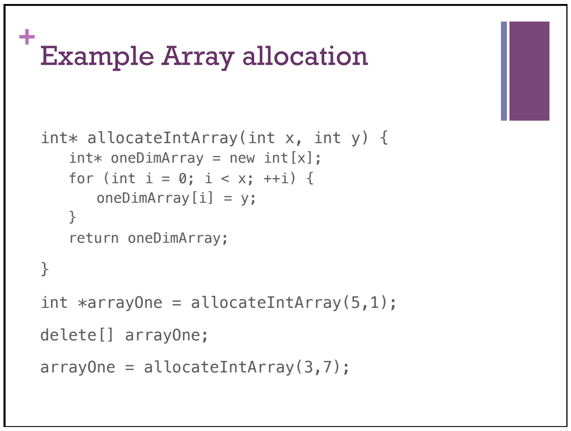


+ Example allocation

- `int* iPtr = new int;`
- `int x = 10;`
- `*iPtr = x * x + 5;`
- `iPtr = new int;`
- `*iPtr = x;`
- `x = *iPtr − x;`
- `delete iPtr;`

Trace the above code, and draw a memory diagram.

How do I get to the memory that holds the value 105?



+ Example Array allocation

```
int* allocateIntArray(int x, int y) {
   int* oneDimArray = new int[x];
   for (int i = 0; i < x; ++i) {
      oneDimArray[i] = y;
   }
   return oneDimArray;
}
int *arrayOne = allocateIntArray(5,1);
delete[] arrayOne;
arrayOne = allocateIntArray(3,7);
```