C++ Pointers, Memory

Readings- Week 7

Pointers and Dynamic memory, stack verses heap

Intro to pointers

Excellent Reference

More Pointers

- A way to allocate/manage memory on the heap
- A way to rapidly iterate over arrays
- For C use malloc and free
- For C++ use new and delete

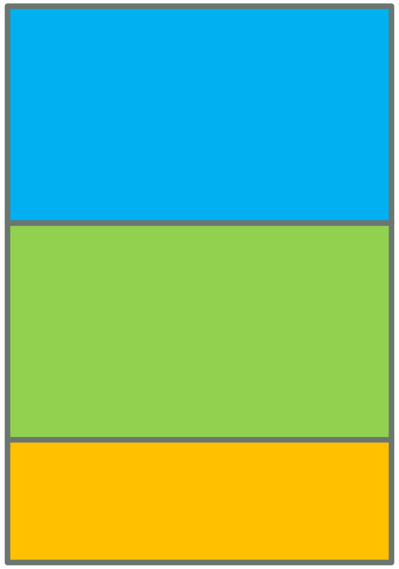
If up to you

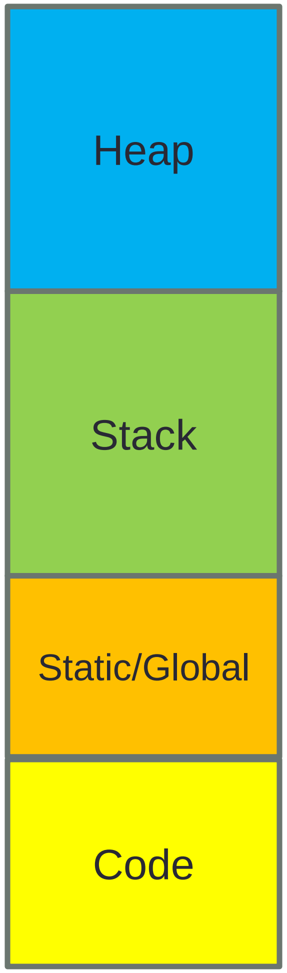
- For C
  - You have to use pointers
- For C++ … Caution
  - Pointers are the source of many, many bugs
  - Use Standard Library instead, it allocates and manages heap memory for you

Pointers – Correct (ish)

Throw an exception an things go poorly

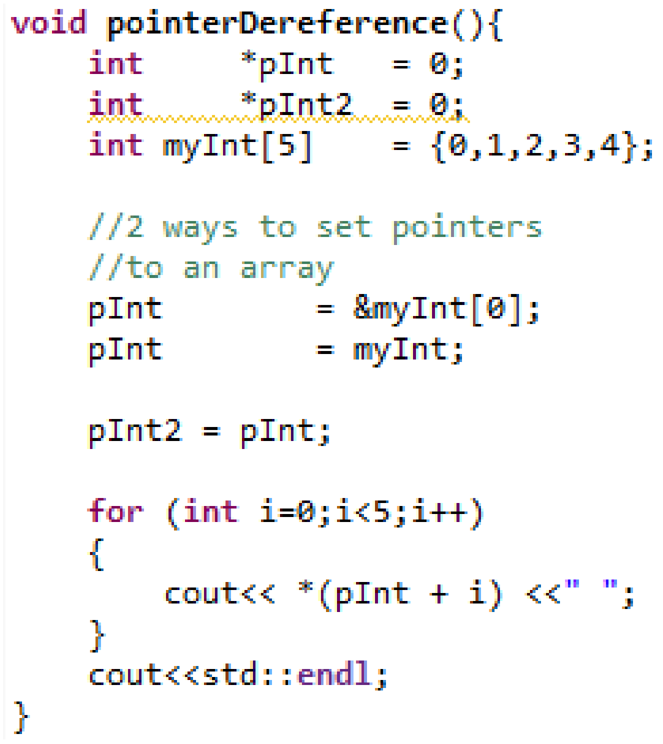
```
const int MY_SIZE = 10;

bool dynamic_good() {
    int *p = new int[MY_SIZE];

    //do some work

    //free if allocated
    if (p)
    {
        delete[] p;
        p = 0;
    }
}
int main()
{
    dynamic_good();
    return 0;
}
```



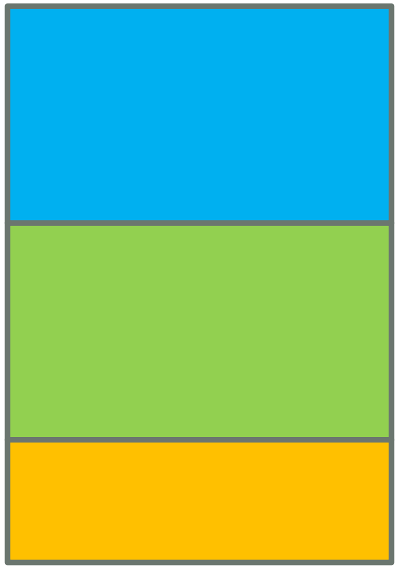


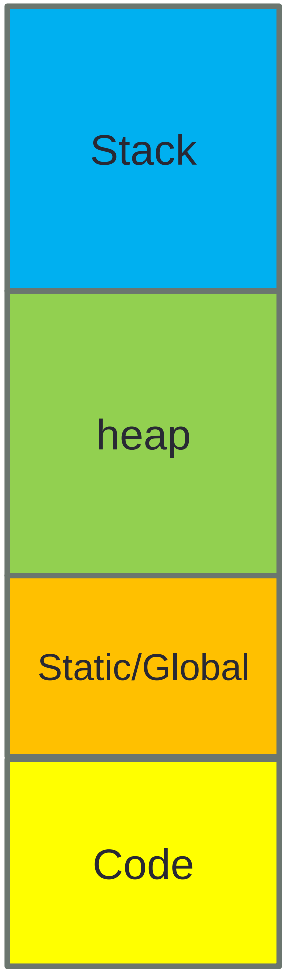
Pointers - Dereference

Do you need to free anything?



```
void pointerDereference(){
    int      *pInt    = 0;
    int      *pInt2   = 0;
    int myInt[5]      = {0,1,2,3,4};

    //2 ways to set pointers
    //to an array
    pInt          = &myInt[0];
    pInt          = myInt;

    pInt2 = pInt;

    for (int i=0;i<5;i++)
    {
        cout<< *(pInt + i) <<" ";
    }
    cout<<std::endl;
}
```

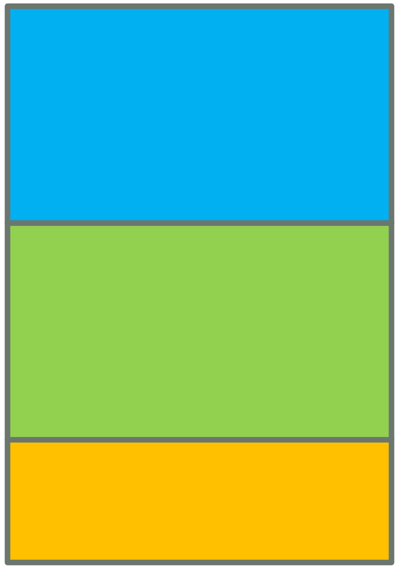


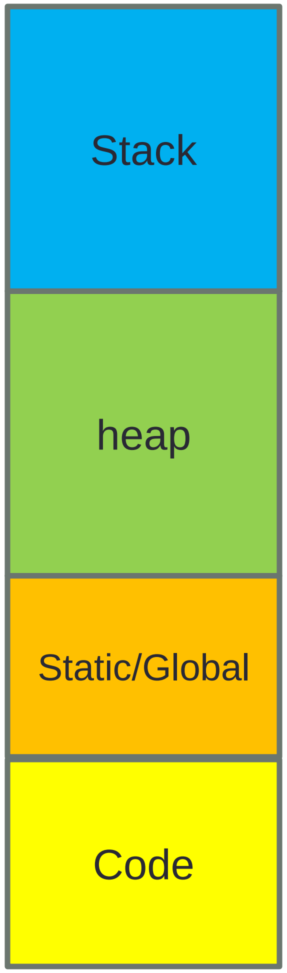


Pointers – Dangling

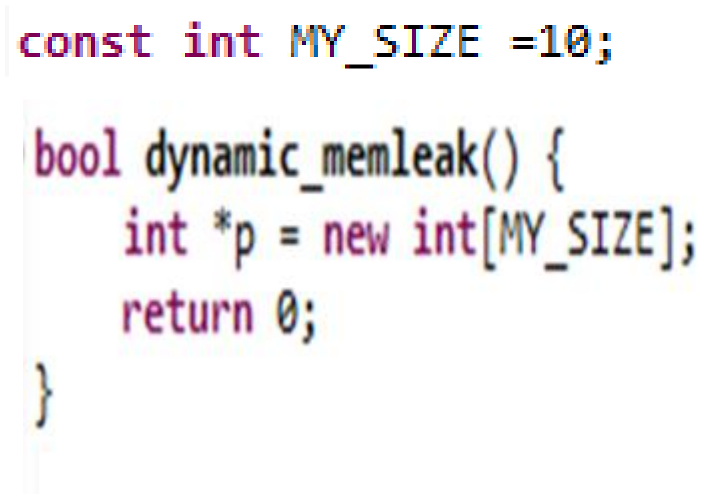
```
int *p = new int[MY_SIZE];
if (!p)
    return false;

int *p2 = p;

if (p)
{
    delete [] p;
    p=0;
}

//what about p2?
```

This check not needed with modern compilers

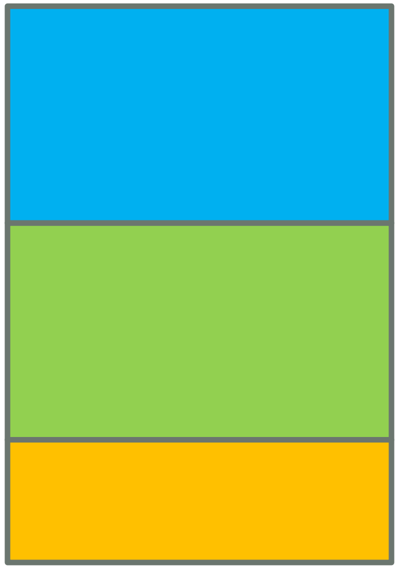


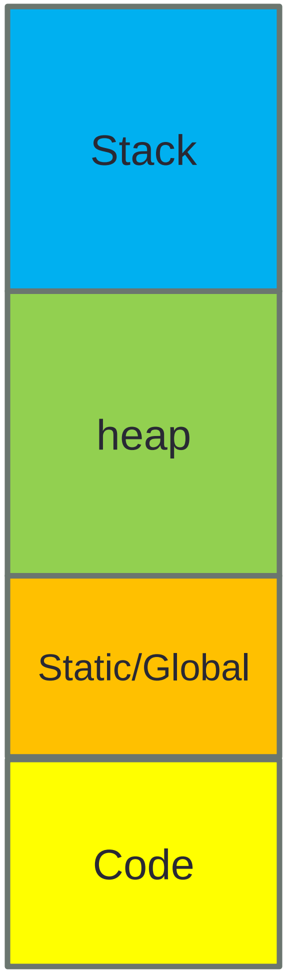


Pointers – Memory Leak

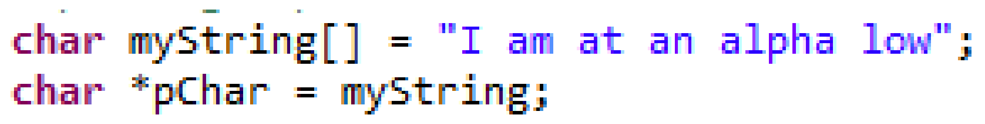


```
const int MY_SIZE =10;

bool dynamic_memleak() {
    int *p = new int[MY_SIZE];
    return 0;
}
```

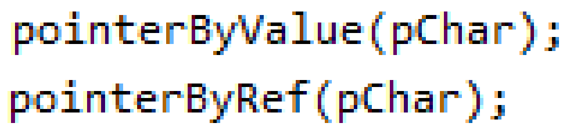




Passing Pointers - review

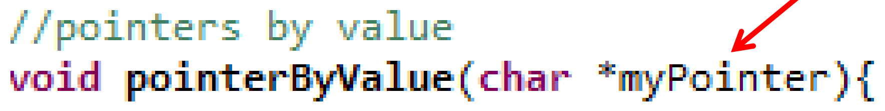


```
char myString[] = "I am at an alpha low";
char *pChar = myString;
```

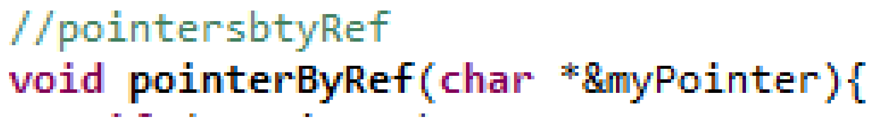


```
pointerByValue(pChar);
pointerByRef(pChar);
```

Remember to verify
That myPointer !=0
Before you dereference



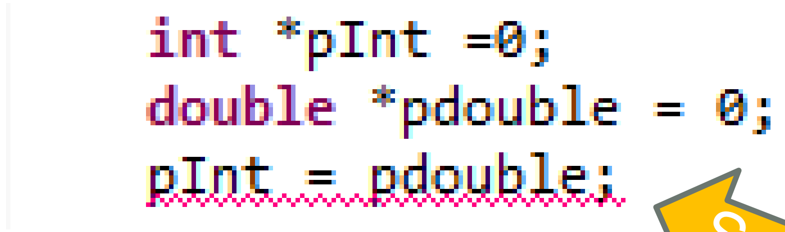
```
//pointers by value
void pointerByValue(char *myPointer){
```



```
//pointersbtyRef
void pointerByRef(char *&myPointer){
```

Pointers - different types

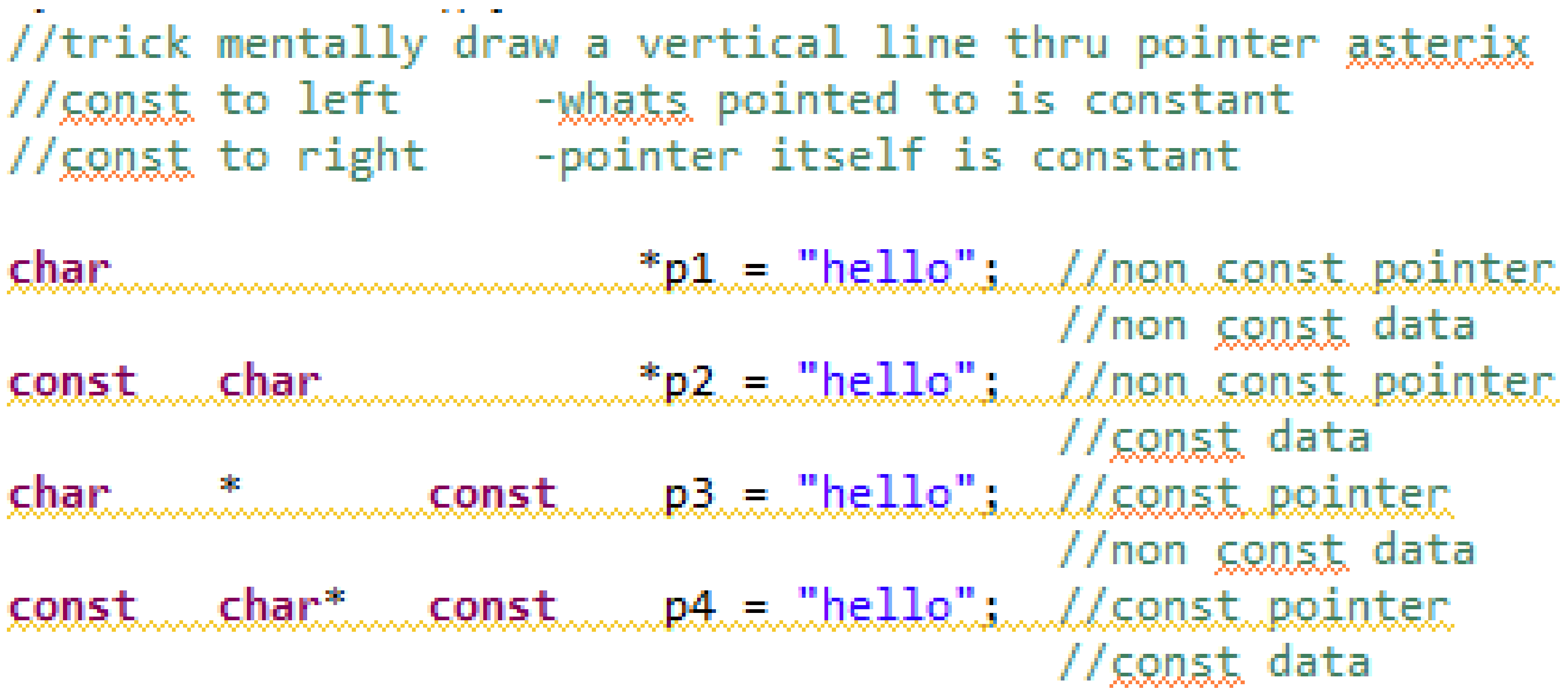
- Pointers to different types are different
- Cannot (for the most part) assign 1 to another



```
int *pInt =0;
double *pdouble = 0;
pInt = pdouble;
```

Syntax Error

Pointers and const



```
//trick mentally draw a vertical line thru pointer asterix
//const to left     -whats pointed to is constant
//const to right    -pointer itself is constant

char                       *p1 = "hello";  //non const pointer
                                           //non const data
const   char               *p2 = "hello";  //non const pointer
                                           //const data
char    *       const       p3 = "hello";  //const pointer
                                           //non const data
const   char*   const       p4 = "hello";  //const pointer
                                           //const data
```

Pointer tip

- If you create something using new[]
- You must delete using delete[]

- If you create something using new
- You must delete using delete

- **//Example**

```
int *p=new int[10];
delete p; //undefined behaviour, sometimes OK sometimes
          //not
```

Conclusions

- Pointers are dangerous
- Please study this lecture, readings (especially intro one) and the example program 07_PointersMemory
- We will see more pointers as we start on objects.