Dynamic Memory: What on Earth is it, and Why do We Care?

(Slide 1 of 5)

A short review to prompt any questions

Dynamic Allocation: *The assignment of memory to a variable during program execution, as opposed to during compilation.*

So how do we do this?

Step 1: DESIGN

Always think carefully about how you're going to use dynamic memory before you do it! Design everything on paper first—the compiler won't check this for you.

Step 2: Creating a new object using dynamic memory

```
myType* myItem = new myType;
```

Note that *new* returns a pointer to a new object in memory. That is, it goes off and allocates the space for your data, and then returns the address so you can find the object in memory, use it, and most importantly, GET RID OF IT WHEN YOU'RE DONE.

Step 3: Deleting, or deallocating dynamic memory

```
delete myItem;
```

Note that delete follows the pointer to its destination and gets rid of the memory allocated for whatever object is there. Do NOT try and dereference the pointer after using delete!

Dynamic Memory: What on Earth is it, and Why do We Care?

(Slide 2 of 5)

An Example of Dynamic Memory in Action: the ***Vector*** *Class*

```
// Vector.h
// Specifies a class containing a
// resizable array of ints.

#IFNDEF _VECTOR_H_
#DEFINE _VECTOR_H_

class Vector {

        public:

        // our constructor
        Vector( );

        // add an item to our vector, resizing
        // if there isn't enough room.
        void addItem( int x );

        // find an item in our vector.
        int findItem( int loc );

        private:

        // current number of items in the vector
        int currentSize;

        // current size of the vector
        int totalCapacity;

        // our data, a pointer because we are going to
        // dynamically allocate it.
        int* data;
};
```

Dynamic Memory: What on Earth is it, and Why do We Care?

(Slide 3 of 5)

```
// Vector.cpp

#include “Vector.h”

Vector::Vector( ){
        totalCapacity = 10;
        currentSize = 0;
        data = new int[totalCapacity];
}

void Vector::addItem( int x ) {
        // is there room?
        if ((currentSize+ 1) << totalCapacity) {
                currentSize++;
                data[currentSize] = x;
                return;
        }
        // there isn’t room, so we need to resize the array.
        // be VERY careful here!
        else {
                totalCapacity = totalCapacity * 2;
                int * newData = new int[newCapacity];
                for (int i = 0; i < currentSize; i++)
                        newData[i] = data[i];
                delete [] data;
                data = newData;
                currentSize++;
                data[currentSize] = x;
                return;
        }
}

void Vector::findItem( int loc ) {
        return data[loc];
}

Vector::~Vector( ){
        delete [] data;
}
```

Dynamic Memory: What on Earth is it, and Why do We Care?

(Slide 4 of 5)

Dynamic Memory Could Kill Us ALL!

Well, not really, but there are some gotchas to watch out for. . .

Gotcha #1: Memory Leaks

Consider the following chain of commands:

```
#include “Vector.h”

Vector* v1 = new Vector;
Vector* v2 = new Vector;
v1->addItem(10);
v1 = new Vector;
v1->addItem(21);
```

What's wrong with this scenario?

How about this?

```
#include “Vector.h”

Vector* v1;
Vector* v2;
v1 = new Vector;
v1->addItem(3);
v2 = new Vector;
v2 = v1;
v2->addItem(4);
```

One more. . .

```
void change ( int* dynamicArray ) {
        int * newArray = new int[30];
        delete [] dynamicArray;
        dynamicArray = newArray;
}

int main ( ) {
        int* dynamicArray = new int[20];
        change(dynamicArray);
        return 0;
{
```

Dynamic Memory: What on Earth is it, and Why do We Care?

(Slide 5 of 5)

Gotcha #2: Dangling Pointers

Consider the following chain of commands:

What's wrong here?

```
#include “Vector.h”

Vector* v1 = new Vector;
Vector* v2 = new Vector;
v1->addItem(10);
delete v1;
v1->addItem(21);
```

How about this one:

```
int* change ( ) {
        int * newArray = new int[30];
        return newArray
}

int main ( ) {
        int* dynamicArray = change( );
        dynamicArray[0] = 2;
        return 0;
{
```