CSCI 2270
Data Structures and Algorithms
Lecture 4—Memory part 3

Elizabeth White
elizabeth.white@colorado.edu
Office hours: ECCS 128 or ECCS 112
Wed 9:30am-11:00am
Thurs 10:00am-11:30am

Administrivia

- HW1 posted
  - It will be due Sunday, Feb. 2nd
  - We'll cover function parameters and classes Friday
- Read pp.117-133 (skip recursive part) for hw1
- Skim pp.47-64 for description of general bags
  - We'll reinforce the important parts in lecture
- No lab this week; lab resumes next week
- Clicker scores have posted for Wednesday
  - Make sure to set the room frequency (AB)
  - Register your clicker

Floating point numbers

Like integers, computers store floating point numbers in binary form. Consider these little numbers:

Decimal	Binary
0	0
0.5	0.1
0.25	0.01
0.125	0.001

For binary numbers,

a 1 in the first place after the decimal (.) is 2^{-1},
and a 1 in the next place is 2^{-2},

and so forth. All still powers of 2.

Floating point numbers

Some familiar numbers don't work out nicely in binary.

Decimal	Binary
0	0
0.1	0.000110011…
0.333…. (= 1/3)	0.010101…
0.2	0.00110011…

Floating point numbers

Which of these binary numbers would be equal to the decimal number 0.0625?

A) 0.0001
B) 0.000110011…
C) 0.1
D) 0.333….
E) 0.111…

Floating point numbers

Which of these decimal numbers will store in binary form without infinite repeats (…)?

A) 0.7
B) 0.375
C) 0.75
D) A and C
E) C and B

Why should we care?

http://sydney.edu.au/engineering/it/~alum/patriot_bug.html

Time step: 0.1 sec

Accumulated error after 100 hours: 0.34 sec

Error in missile position: 690 m

Boom!

Back to arrays

```
void bubbleSort(int theArray[], int n)
```

Suppose you have to write code that sorts an array of n integers, like the one above. This is a normal thing to do in this class later on. (Don't worry about the specifics of sorting yet.)

Now, suppose you want to adapt this code to sort unsigned ints, or doubles. Normally, *you would need to write 3 different versions of the sorting code* to sort these 3 different types of numbers (int, double, or unsigned int). Yuck.

Template class sorting

```
template<class ItemType>
void bubbleSort(ItemType theArray[], int n)
```

In C++, instead of tying ourselves down to a type, we can define a generic data type (ItemType) and write our code to operate on this ItemType. Note that this lets us write the sorting code exactly once!

When we run this code in a main program, we specify the actual data type in place of the generic ItemType:

```
int main() {
      string a[5] = {"Z","X","R","K","F"};
      bubbleSort(a, 5);
}
```

Template class arrays

```
template<class ItemType>
void bubbleSort(ItemType theArray[], int n)
```

When we run this code in a main program, we specify the actual data type in place of ItemType:

```
int main() {
      string a[5] = {"Z","X","R","K","F"};
      bubbleSort(a, 5);
      int b[8] = {4,-5,9,-13,7,-31,2,1};
      bubbleSort(b, 8);
      double c[3] = {8.6, 5.7, 8.3};
      bubbleSort(c, 3);
}
```

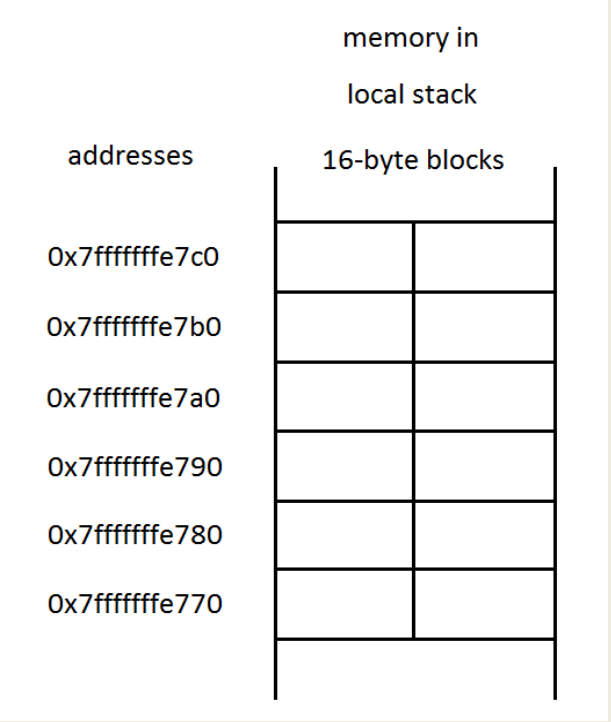
Who can be an Itemtype?

```
template<class ItemType>
void bubbleSort(ItemType theArray[], int n)
```

To be a valid ItemType, the base data type must know how to do comparisons like less than (<). That allows us to sort arrays of strings, or ints, or doubles, or unsigned ints. So any base type with a < operator will be ok here. (This is because we assume that bubbleSort's using < to do the sorting.)

Compiling generic classes

```
template <class ItemType>
void bubbleSort(ItemType theArray[],
int n)
{
        ....
        ItemType i[3];
}
```



What's the problem with this generic ItemType for the compiler?

How can we compile the code, then?

```
template<class ItemType>
void bubbleSort(ItemType theArray[], int n)
```

If we compile the C++ code that has the main() function, and that main() C++ code includes the template C++ code (via #include), then the template code will compile along with the main code.

For hw1, which is a template class, you do something similar: compile ArrayBagTester.cxx, and ArrayBag.cxx will compile too.

Friday: C++ classes

Classes can capture more complicated behavior

Describe a video game character

- Health? Wealth? Weaponry? Remaining lives?
- Location, heading?
- Destination to reach?

Describe a particular level of the video game

- Map of territory, with doors & walls
- Monsters
- Traps, or goodies

How would these two descriptions interact?

- What if another character is in the same area as we are?
- What if another character has found the goodies?