Designing Abstractions

Fundamental Question

How do our tools work?

Classes

- **`Vector`**, **`Stack`**, **`Queue`**, **`Map`**, etc. are **classes** in C++.
- Classes contain
    - An **interface** specifying what operations can be performed on instances of the class.
    - An **implementation** specifying how those operations are to be performed.
- To define our own classes, we must define both the interface and the implementation.

Random Bags

- A **random bag** is a data structure similar to a stack or queue.
- Supports two operations:
  - **Add**, which adds an element to the random bag, and
  - **Remove random**, which returns and removes a random element from the bag.
- Has several applications:
  - Random maze generation
  - Shuffling decks of cards.

Let's Code it Up!

Classes in C++

- Defining a class in C++ (typically) requires two steps:
  - Create a **header file** (typically suffixed with `.h`) describing the class's member functions and data members.
  - Create an **implementation file** (typically suffixed with `.cpp`) that contains the implementation of all the class's member functions.
- Clients of the class can then include the header file to use the class.

Midterm Logistics

- **Midterm room assignments will be changing.**
- **You should receive an email about this by the end of the night.**
- My sincerest apologies – this is the first time this has ever happened. We'll do our best to ensure this doesn't happen again.

Language Philosophy

- Every programming language exports some set of **primitives**:
  - Primitive data types (**`int`**, **`char`**, etc.)
  - Functions
  - Classes
  - etc.
- We can use those primitives to construct a larger set of primitives:
  - **`Vector`**, **`RandomBag`**, etc.

Where Does it Stop?

- The collections we've been using are not primitives in C++; they are defined in terms of other language features.
- Understanding those features will let us analyze their efficiency.
- Understanding those features will let us build other interesting abstractions.

All About Memory

What is Memory?

- All variables and objects in C++ need somewhere to live inside the computer's memory.
  - This is RAM, by the way, not disk space.
- Whenever an object is created, space needs to be reserved for it.
- Where does this memory come from?

Memory So Far

- So far, you have seen two types of variables:
- **Local variables** declared inside a function.
  - Space is reserved for these variables when the function is called.
  - Space is reclaimed from these variables when the function call ends.
- **Global variables / constants** declared outside a function.
  - Space is reserved for these variables when the program stars up.
  - Space is reclaimed from these variables when the program exits.

Getting Space

```
int main() {
    Vector<int> values;

    int numValues = getInteger("How many?");
    for (int i = 0; i < numValues; i++) {
        values += i;
    }
}
```

Getting Storage Space

- How do the **`Vector`**, **`Stack`**, **`Queue`**, etc. get space to store all the elements that they hold?
- C++ code can request extra storage space as the program is running.
- This is called **dynamic memory allocation**.

Good luck on the exam!