cis1.5 spring2009 lecture IV.1

today we are going to talk about...

- what are arrays and why to use them
- integer arrays

arrays

- arrays are used to hold sets of related types of data
- the data could be integers or doubles or booleans
- the data could also be characters;
  arrays of characters are special arrays called *strings*
  we'll talk about those another day
- today, we'll focus on arrays that store numbers (e.g., `int` or `double`)
- common things to do with numeric data stored in arrays:
  - find the largest (or smallest) element
  - add up the elements
  - compute the average of the elements
  - count the number of elements with some feature

what is an array?

- you can think of an array as a set of variables of the same data type, which are grouped together and all use the same identifier (name).
- just as

  `int a;`

  declares one integer variable with the name `a`, then

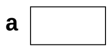
  `int b[5];`

  declares an *array* of 5 integers, with the name `b`.
- the square brackets `[ ]` are the crucial bit of syntax, telling the compiler it is dealing with an array

- whereas

  `int a;`

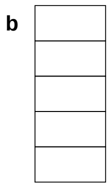
  reserves space for one integer in memory and associates the name `a` with it:



the declaration

`int b[5];`

reserves space for five integers in memory right next to one another.



- elements of the array b are just integers, and we can do exactly the same things with them that we can do with integers
- the only difference is how we *address* (i.e., refer to) them
- while we can assign a value to a by:

  ```
  a = 5;
  ```

  to do the same to one of the *elements* of b, we have to specify which element it is.

  for example:

  ```
  b[1] = 5;
  ```

- all of the following are legal operations:

  ```
      b[1] += 2;
      b[2] =  7 % 3;
      b[3] =  b[2] - 5;
      b[4] =  b[1]/b[3];
  ```

- one thing to be careful of is the limits on the *index*, that is the number inside the square brackets [ ]
- the first element of an array always has index 0
- so the first element of b is:

  ```
  b[0]
  ```

  and, since b has 5 elements, the last element of b is:

  ```
  b[4]
  ```

- in other words, the last *index* is *the length of the array minus 1*
- this type of counting (from 0 to length-1) is standard in C, C++ and Java and many other computer languages

- arrays are useful when you want to store lots of data in memory
- if I want to use 3 integers in my program, then I would just declare 3 different integer variables
- however, if I wanted to use 30,000 integers in my program, it would be a lot easier to use an array than to declare 30,000 different integer variables!
- arrays also go very nicely with for loops

- here is some sample code that declares an integer array of 100 values and stores random numbers in the array:

  ```
  int a[100];
  int i;
  for ( i=0; i<100; i++ ) {
    a[i] = rand();
  } // end for
  ```

- once the data is stored in the array, we can do all kinds of stuff with it
- for example, we can print out the values in the array:

```
for ( i=0; i<100; i++ ) {
  cout << a[i] << endl;
} // end for
```

- or we could print out the values in reverse order:

```
i = 99;
while ( i > 0 ) {
  cout << a[i] << endl;
  i--;
} // end while
```

- another thing we can do is to add up all the values in the array:

```
int sum;
sum = 0;
for ( i=0; i<100; i++ ) {
  sum += a[i];
} // end for
cout << "the sum of all the values in the array is: "
     << sum << endl;
```

- and another thing we could do is to find the smallest value in the array:

```
int smallest;
smallest = a[0];
for ( i=1; i<100; i++ ) {
  if ( a[i] < smallest ) {
    smallest = a[i];
  }
} // end for
cout << "the smallest value in the array is: "
     << smallest << endl;
```