CS120
Lab 9

Classes (summary), pointers and arrays.
Fabian Mathijssen

Classes (summary)

- A class is:
  - Class definition (header file or inline)
    Remember: include header files with **<>** *or “”*
  - Source code file
- Classes may model anything in real life.
- Classes are blueprints!
  - Usually useless by themselves
  - Good for making objects -> have objects do stuff

Pointers (1/4)

- Pointers are memory addresses, so numbers.
- Written as *, pronounced “pointer” or “star”.

- Pointers have types!
  ➔ Primitive types: **int***, **float***, **char***, etc.
  ➔ Object types: **Chair***, **Monitor***, **Book***, etc.

- The size of a pointer depends on the operating system you’re using. In a nutshell, 64-bit computers have 64-bit memory addresses, so pointers are 8 bytes large.
  Tip: use sizeof(variableName) to check size of a variable!

Pointers (2/4)

- Create pointer: **int *i;**
  The pointer **i** doesn't point to anything right now, so in other words, the memory address **i** points to doesn't contain a value.
- Dereference pointer: ***i = 28;**
  Now the memory address **i** points to contains value 28.
- Get a variable's address: **int k = 26;**
  i = &k;
  Now the pointer **i** points to the address of **k**, which means **i** points to an address that contains value 26.

- *& is pronounced as "address of".*

Pointers (3/4)

You can create a pointer to a pointer!

```
int **k; //a reference to a memory location that holds a
         //reference to a memory location that holds an
         //integer value!
```

```
**k = 17;
int m = **k;  //Deferences the pointer to k and dereferences
              //the pointer to the dereferenced value.
              //Puts the dereferenced value into m.
```

Pointers (4/4)

As you may see, one can certainly overuse pointers!

int **********m;

****************m = 128;**

Arrays (1/4)

- An array is a sequence of elements.
- An array has a number of positions that cannot be changed when set.

  Ex.: **char name[6] = {‘F’, ‘a’, ‘b’, ‘i’, ‘a’, ‘n’};**

‘F’	**‘a’**	**‘b’**	**‘i'**	**‘a’**	**‘n’**

- An array is really a pointer to the first element in a sequence of elements!
  We’ll overlook this for now. ☺

Arrays (2/4)

- Get the element at position **i** from the array like so:

```
int longnumber[6] = {9, 7, 3, 4, 5, 1};
int secondDigit = longNumber[1]; // = 7
```

Note: an array with *n* positions has positions numbered 0 through *n*-1.

Arrays (3/4)

- Go through an array with for loops to get each individual element:

```
int longnumber[6] = {9, 7, 3, 4, 5, 1};

for (int i=0; i<6; i++) {
    cout << longNumber[i];
}
```

- *Note: store the length of the array in a variable so you know how many iterations the* for *loop must have!*

Arrays (4/4)

- You can have an array of arrays, a.k.a. a 2-dimensional array.
- *Note: Each position is a pointer to the first element of another array.*
- Make a 2-D array with **x** by **y** positions like this:
  int twoDArray[y][x];
- Retrieve an element from this array at position 3 by 5 like so:
  int retrievedElement = twoDArray[3][5];

Last note on arrays...

- Trick question: how do you get the size (nr of positions) of an array?
- How do you get the size of a 2-D array?

- *Hint: use sizeof()!*

Retrospective

- How hard do the week 1-3 labs seem now? ;-)