In-Class Work: C++

1. Copy & Compile & Run

This problem guides you through the first steps of writing and running a program:

Log into linuxremote1 (redo the steps 1.-3. of the Unix/Linux Exercise). Get into your capstone directory. Next make a directory for C++ sample files, so

`mkdir C++samples.dir`

Get into this directory (using `cd`) and copy all C++-sample programs by typing

cp ~kvollmay/classes.dir/capstone_s2009.dir/unix_C++_intro.dir/C++sample*.cc .

Also copy some necessary data-file

cp ~kvollmay/classes.dir/capstone_s2009.dir/unix_C++_intro.dir/birthdays.data .

Look at C++sample_inout.cc with

`emacs C++sample_inout.cc &`

This is the text file ("source code") of a program. This textfile needs to be converted to a format which the computer will understand. To do so you "compile" the source code (text file):

`g++ -o C++sample_inout.out C++sample_inout.cc`

This creates a file C++sample_inout.out (the "executable file") which you can now use to run with

`C++sample_inout.out`

You will be asked to type in your first and last name separated by blanks, do so and hit enter and then answer the next question. Check in the source code that you understand what exactly the program was doing.

2. Input/Output & Repetitions

2a. Write a program that reads in (from screen) the name of a fruit (e.g. banana) and its price (e.g. 0.5), and prints out for this example: `One banana costs $0.5`.

2b. Copy ~ kvollmay/classes.dir/capstone_s2009.dir/unix_C++_intro.dir/groceries.data into your working directory. Write a program which reads in 10 groceries and their prices from the file groceries.data using fstream and prints a sentence for each item as in 2a.

2c. Same as 2b. but with $<$ on the command line. To do so use `cin` in the program and run the program with *executable.out* `< groceries.data`
where *executable.out* is the name of your executable file.

2d. Next you learn how you will share your programs with others. Let's assume you had called your program for 2c "C++2c.cc." Make this file readable (accessible to be copied) for others by typing

`chmod a+r C++2c.cc`

Whenever I will ask you to make your programs available for others (e.g. at the next homework assignment) please copy your programs into your ~ /share.dir and make the corresponding program readable with `chmod a+r` *filename.cc*

3. Repetitions Read in the 10 groceries (item and price) from the file groceries.data. Print out each item and its price, the sum of the prices, the sales tax (use 5%, i.e. sum*0.05), and the total cost (= sum + tax).

4. Factorial Write a program which calculates $N! = 1 * 2 * 3 * \ldots * N$ and prints out N and $N!$ for $N = 1, 2, \ldots, 20$. What happens for $N \geq 14$?

In-Class Work: C++

3. Repetitions (Groceries) First copy a file with data of 10 groceries (item and price)
cp ~kvollmay/classes.dir/capstone_s2009.dir/unix_C++_intro.dir/groceries.data .
Now write a program that reads from the file groceries.data. Print out each item and its price, the sum of the prices, the sales tax (use 5%, i.e. sum*0.05), and the total cost (= sum + tax).

4. Spring Break (Decisions)

Read in a date in 2009 (as two integers) and check if it is during spring break (March 7 - March 15).

5. Traffic Model (Decisions & Repetitions

Copy the file
~ kvollmay/classes.dir/capstone_s2009.dir/unix_C++_intro.dir/road.data
into your working directory. The file contains 100 integers describing a road of 100 lattice sites at a certain time. "-1" mean an empty site and a number $>= 0$ means a car with the corresponding speed in mi/h.

5a. How many cars are on this road?

5b. How many cars have a speed larger or equal 30 mi/h?

6. Time (Functions)

The following task was once needed by a student whose project was the prediction of triathlon athletes performance based on their performances in the past.

6a. Write a program which reads in some time length as 3 integers for the number of hours, minutes and seconds and which converts this into the total amount of seconds (for example 3 h 4 min 2 s = (3*3600 + 4*60 + 2)s = 11042 s). Print out the result.

6b. Now do the task of the convertion not in main but in a user-defined function.

6c. Write a program which reads in a total number of seconds (integer) and uses a user-defined function to convert the seconds to hours, minutes and left over seconds. Print out the result for hours, minutes and seconds. For example 11042 s = 3 h, 4 min and 2 s.
Hint: You get the hours by using integer division (division of two integers), e.g. (# hours) = 11042/3600 = (total # seconds)/3600 and you get the left over seconds with the modulo function %, e.g. (# left seconds) = 11042 % 3600 =242 . Now you can calculate how many minutes are in the left over 242 seconds.

Solutions:
~kvollmay/classes.dir/capstone_s2009.dir/unix_C++_intro.dir/C++2a.cc etc.

In-Class Work: C++

7. Traffic Model (1d-Array) As you did for the in-class work 5., copy the file
~ kvollmay/classes.dir/capstone_s2009.dir/unix_C++_intro.dir/road.data
into your working directory. The file contains 100 integers describing a road of 100 lattice sites at a certain time. "-1" mean an empty site and a number $>= 0$ means a car with the corresponding speed in mi/h.

7a. Either use your program for the in class work 5a. or copy the solution program
~ kvollmay/classes.dir/capstone_s2009.dir/unix_C++_intro.dir/C++5a.cc
into your working directory. The program reads in the road data and determines how many cars are on this road. Change the program such that the road data are first all read into an array and then this array is used to determine the number of cars.

7b. Now change the program such that the number of cars is determined via a function.

7c. (Tool For Fun) You can look at the street with
`cat road.data | DynamicLattice -nx 100 -ny 1 -matrix`
or with
`DynamicLattice -nx 100 -ny 1 -matrix < road.data`

8. Contrast Filter (2d-Array)

Copy into your working directory
~ kvollmay/classes.dir/capstone_s2009.dir/unix_C++_intro.dir/pic1.data
This file contains a 10x10 matrix. Have a look at this matrix (picture) with `DynamicLattice` and with `cat`. Hidden in this picture is some pattern, which you can see if you make a "filter", i.e. if you change the picture to a picture which gives you more contrast: Write a program which reads in this 10x10 matrix and prints out into the file ("pic1contrast.data") a different 10x10 matrix which replaces every number < 0.5 with -1.0 and every number ≥ 0.5 with $+1.0$. Look at piccontrast.data with DynamicLattice.

9. Population Dynamics (if time)

One of the simplest models for population growth of one species (e.g. humans) is the so called "logistic map." You describe the population with one number n and say from year t to year $t+1$ that n changes:

$$n_{t+1} = an_t(1 - n_t) \tag{1}$$

where a is a constant and $0 \geq n \geq 1$ (that means the number of people N has been devided by the maximum number of people $n = N/N_{\max}$). If you rewrite the equation to $n_{t+1} = an_t - an_t * n_t$ you see that from year t to $t+1$ you get an_t more people (because they are born) and you get $an_t * n_t$ fewer people (because they die). Write a program which starts with $n(t=0) = 0.5$ and then you do 50 timesteps $t = 1, \ldots, 50$, so a loop which does Eq.(1) again and again. Print for each time step t and n_t (and use the `endl`).
Run your program for $a = 2.5$ (let's say your program is "logmap.out" and look at the output with
`logmap.out | xgraph -m`
Then run your program with $a = 2.7, a = 3.2, a = 3.5$ and $a = 3.7$. Describe your results.