January 18th

* $ ls – list of files
* $ script
* $ ls
* $ a.out
* A + b \* c
* A b c \* +
* ^D
* $
* ^D
* End of command will create the typescript file. Check ls to make sure the file is there. Then send that file to the printer
* Debugging on gedit and terminal (gdb) gnu debugger
  + Compile program with
    - $ g++ -g expr.cpp if no errors you will get your prompt out
    - $ gdb a.out to run with debugger
    - > break 14 – breaks the compiler at line 14 to check values
    - > run will run the program
    - Will break at line 14
    - > p I – print I to check value
    - > c will continue the program for example if your line is in a for loop
    - > n will run till the next statement
    - > help will reply with all the commands gnu debugger has.
* Rename youre a.out
  + $ g++ expr.cpp -o (then your name ie expr)
* Lab 2 analysis
  + Bobble Sort
    - Array goes from bottom to top (ie 0 – 1 – 2.... 7)
    - Compare the two adjacent components to see if they are out of sequence. And swap if they are. Highest value will bobble to the top. Each run will go to one lower then the previous leaving the highest value of each run at that current top
    - Algorithm
      * For (I =n-1; I >0; I--)
        + For (j = 0; j < I; j++)

If (v[j] < v[j+1])

Swap(v[j],v[j+1]);

* + - * O(n^2) algorithm
  + Insertion sort
    - Array left to right
    - If partial of the array is sorted properly and you don’t know what you whats going on after.
    - If the next is lower then the previous you take it out into a temp variable then copy the previous over to where the temp came from. You drag the temp through the list copying each variable that is bigger over one spot till the temp is bigger then the comparison and it replaces that spot.
    - Algorithm
      * For (I =1; I < n; I++) start after spot 0 because its already sorted
        + For (elem = v[I], j = I-1; j >=0 && elem < v[j]; j--)

V[j+1] = v[j];

* + - * + V[j+1] = elem; (outside the second for loop but inside the first loop)
    - T(n) = O(n^2) (not = but set membership operator)
    - Ex:
    - #include <cstdlib>
    - Main()
      * Int n;
      * Cout << "Enter a number of numbers in a list to sort" << endl;
      * Cin >> n;
      * Vector<int> v(n);
      * // generate random numbers [0, 1000000)
      * Srand(time(0)); (just once before the first call to rand)
      * For (int I = 0; I < n; I ++)
        + V[I]= rand()%1000000;
      * Then pass the vector to the sort mechanisms, but need to pass by reference.
      * Void select\_sort(vector<int> &v) (Passed by reference with &)(header of the function of selection sort)
    - Timing the program
      * $ time a.out
        + Returns 3 times

Real time – from pressing enter till the next prompt came

User time – time for the program to execute

System time – time for the os to run the program

We need the user time

We create a table for the user times with n incrementing from 1000 to 10000.

|  |  |  |  |
| --- | --- | --- | --- |
| n | N^2 | T | (T/n^2) = c |
| 1000 | 1000000 | ? | Constant |
| 2000 | 4000000 | ? | Constant |

* + Create one table for each sort function.
* To avoid having terminal printing all 1000 sorted numbers in the window then you do
  + $ time a.out > keep
* Wants three different programs, one for each selection sort?
* Include files
  + #include <iostream> (<> means files from the system)
    - All the source code from iostream is added to your code. Essentially lengthening the program.
    - Cout and Cin are functions inside iostream.
  + #include "myfile.h"
    - "" for the system to find the files inside the current directory.
    - Every include file from user you need
      * #ifndef MYFILE\_H
      * #define MYFILE\_H
      * -
      * -
      * #endif
* Separate Compilation
  + Main.cpp -
    - #include <iostream>
    - Using namespace std;
    - Int main ()
    - {
      * Void f(); // signature or prototype
      * F();
    - }
  + f.cpp
    - #include <iostream>
    - Using namespace std;
    - F()
    - {
      * Cout << "f() \n" ;
    - }
  + $g++ -c (just compiles without the .exe) main.cpp
    - If no errors it will create main.o (object program, not linked to anything)
  + $g++ -c f.cpp
    - Will create f.o
  + $g++ main.o f.o
    - Will create the a.out
  + $ a.out
* Pointers & Arrays
  + The name of an array is a pointer to the beginning of it.
  + Main(){
    - Int I = 1, \*p; (create a p to point to type int)
    - P = &I; (address of I is set to p. so p is going to I)
    - \*p = 2; (sets whatever p is pointing to, to 2)(changing I from 1 to 2)
    - Cout << I; (prints out 2)}
  + Say the computer assigned I to memory location 200. Since I is an int 200,201,202,203 are all assigned I
  + Pointers are 4 bytes as well. So mem location 1000, 1001, 1002, 1003 are all reserved for p.
  + P = &I makes the p memory locations store the memory location of I. so it would store 200 into it
  + \*p = 2; will go to p which points to location 200 so that itll change the location 200 to the number 2.