Assignment 1 Nathan Paternoster

#include <iostream>

#include <cstdlib>

#include <sys/time.h>

#include <sys/resource.h>

using namespace std;

int random\_in\_range(int a, int b) {

return a+random()%(b-a+1);

}

char random\_letter() {

static string alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

int pos = random\_in\_range(0, alphabet.length()-1);

return alphabet[pos];

}

string random\_string() {

int length = random\_in\_range(3,12);

string result = "";

for (int i=0; i<length; i++)

result += random\_letter();

return result;

}

void randomize(string A[], int n) {

for (int i=0; i<n; i++)

A[i] = random\_string();

}

void swap(string & a, string & b) {

string temp = a;

a = b;

b = temp;

}

bool compare(string & a, string & b) {

if (a>b) return true;

else if (a<b) return false;

}

void selection\_sort(string A[], int size) {

int x = 0;

for (int i=0; i<size-1; i++) {

x = i;

for (int j=i+1; j<size; j++) {

if (compare(A[x],A[j])==true) x = j;

}

swap(A[x],A[i]);

}

}

void insertion\_sort(string A[], int size) {

for (int i=0; i<=size; i++) {

for (int j=i-1; j>0; j--) {

if (compare(A[j],A[j-1])==false) {

swap(A[j],A[j-1]);

}

}

}

}

void bubble\_sort(string A[], int size) {

for (int i=size; i>0; i--) {

for (int j=0; j<i-1; j++) {

if (compare(A[j],A[j+1])==true) swap(A[j],A[j+1]);

}

}

}

double getcputime(void) {

struct timeval tim;

struct rusage ru;

getrusage(RUSAGE\_SELF, &ru);

tim=ru.ru\_utime;

double t=(double)tim.tv\_sec + (double)tim.tv\_usec / 1000000.0;

tim=ru.ru\_stime;

t+=(double)tim.tv\_sec + (double)tim.tv\_usec / 1000000.0;

return t;

}

void main() {

int size;

double starttime, endtime;

string sort;

cout << "How many strings in the array?\n";

cin >> size;

cout << "\n";

string data[size];

randomize(data,size);

for (int i=0; i<size; i++) {

cout << data[i] << " ";

}

cout << "\n\n" << "Insertion, Selection, or Bubble Sort? (Type 'Insertion', 'Selection', or 'Bubble')\n";

cin >> sort;

cout << "\n";

starttime = getcputime();

if (sort=="Insertion" || sort=="insertion") insertion\_sort(data,size);

else if (sort=="Selection" || sort=="selection") selection\_sort(data,size);

else if (sort=="Bubble" || sort=="bubble") bubble\_sort(data,size);

endtime = getcputime();

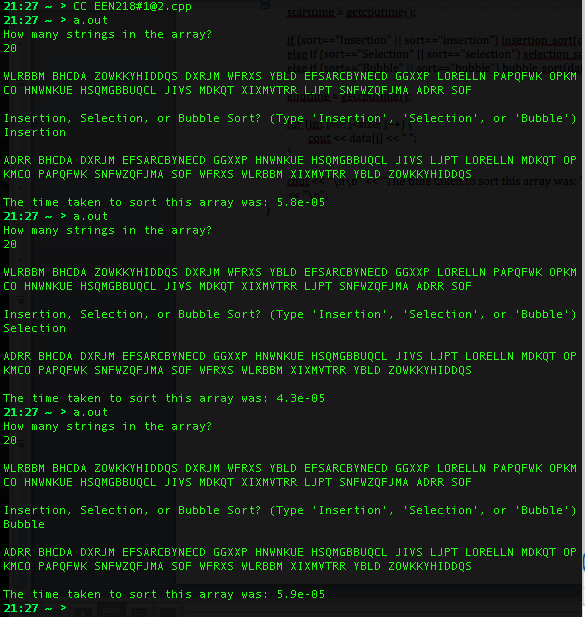
for (int i=0; i<size; i++) {

cout << data[i] << " ";

}

cout << "\n\n" << "The time taken to sort this array was: " << endtime-starttime << "\n";

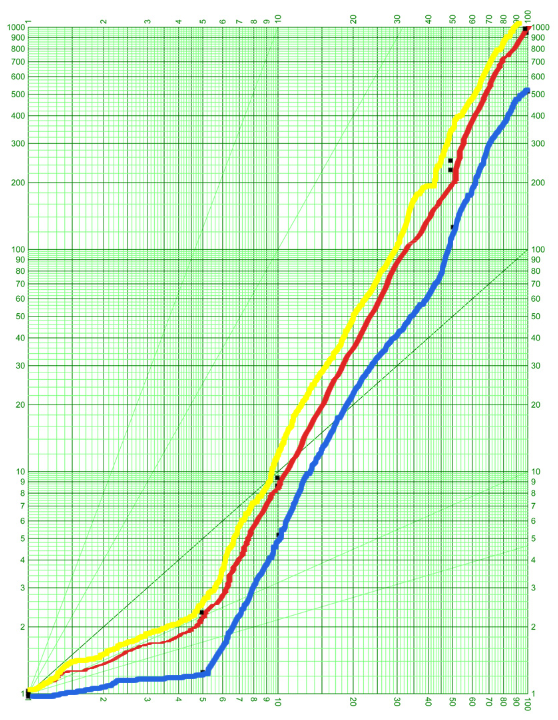
}



|  |  |  |
| --- | --- | --- |
| **Sorting Method** | **Array Size** | **Time (s)** |
| Insertion | 10 | 0.000025 |
| Insertion | 50 | 0.000255 |
| Insertion | 100 | 0.000921 |
| Insertion | 500 | 0.022395 |
| Insertion | 1000 | 0.089846 |
| Insertion | 5000 | 2.324440 |
| Insertion | 10000 | 9.386590 |

|  |  |  |
| --- | --- | --- |
| **Sorting Method** | **Array Size** | **Time (s)** |
| Selection | 10 | 0.000026 |
| Selection | 50 | 0.000156 |
| Selection | 100 | 0.000531 |
| Selection | 500 | 0.012818 |
| Selection | 1000 | 0.052321 |
| Selection | 5000 | 1.315450 |
| Selection | 10000 | 5.268070 |

|  |  |  |
| --- | --- | --- |
| **Sorting Method** | **Array Size** | **Time (s)** |
| Bubble | 10 | 0.000023 |
| Bubble | 50 | 0.000252 |
| Bubble | 100 | 0.000973 |
| Bubble | 500 | 0.023383 |
| Bubble | 1000 | 0.094443 |
| Bubble | 5000 | 2.405460 |
| Bubble | 10000 | 9.688790 |



Yellow – Bubble Sort

Red – Insertion Sort

Blue – Selection Sort

* I graphed the last 4 points from the table of values. The amount of time for data sizes of really small amounts was ignored when working out the equations. The equations are really just rough estimates.

Equations:

Bubble(x) = x5.79

Insertion(x) = x5.57

Selection(x) = x3.16

I found that bubble was the slowest and insertion sort was almost as slow. Selection sort seemed to be significantly faster than either of the other two although not quite quadratic (for my program).