#include <iostream>

#include <vector>

#include <algorithm>

#include <chrono>

#include <cstdlib>

#include <ctime>

#include <cmath>

#include <string>

#include <iomanip>

#include <fstream>

#include <sys/stat.h>

#include <xlsxwriter.h>

using *namespace* std;

using *namespace* chrono;

*long* *long* comparisons;

lxw\_workbook \*workbook = nullptr;

lxw\_worksheet \*worksheet = nullptr;

lxw\_format \*header\_format = nullptr;

lxw\_format \*row\_format = nullptr;

lxw\_format \*separator\_format = nullptr;

*int* current\_row = 1;

*int* current\_col = 0;

*int* block\_counter = 0;

*bool* generate\_excel = true;

vector<*long* *double*> selection\_times, bubble\_times, insertion\_times, merge\_times, quick\_times;

vector<*long* *long*> selection\_comparisons, bubble\_comparisons, insertion\_comparisons, merge\_comparisons, quick\_comparisons;

vector<*int*> input\_sizes;

*void* selectionSort(vector<*int*> *arr*) {

comparisons = 0;

for (*size\_t* i = 0; i < arr.size() - 1; i++) {

*int* min\_idx = i;

for (*size\_t* j = i + 1; j < arr.size(); j++) {

comparisons++;

if (arr[j] < arr[min\_idx]) min\_idx = j;

}

swap(arr[i], arr[min\_idx]);

}

}

*void* bubbleSort(vector<*int*> *arr*) {

comparisons = 0;

for (*size\_t* i = 0; i < arr.size() - 1; i++) {

for (*size\_t* j = 0; j < arr.size() - i - 1; j++) {

comparisons++;

if (arr[j] > arr[j + 1]) swap(arr[j], arr[j + 1]);

}

}

}

*void* insertionSort(vector<*int*> *arr*) {

comparisons = 0;

for (*size\_t* i = 1; i < arr.size(); i++) {

*int* key = arr[i];

*int* j = i - 1;

while (j >= 0) {

comparisons++;

if (arr[j] > key) {

arr[j + 1] = arr[j];

j--;

} else break;

}

arr[j + 1] = key;

}

}

*long* *long* merge(vector<*int*>& *arr*, *int* *l*, *int* *m*, *int* *r*) {

*long* *long* comps = 0;

vector<*int*> L(arr.begin() + l, arr.begin() + m + 1);

vector<*int*> R(arr.begin() + m + 1, arr.begin() + r + 1);

*int* i = 0, j = 0, k = l;

while (i < L.size() && j < R.size()) {

comps++;

if (L[i] <= R[j]) arr[k++] = L[i++];

else arr[k++] = R[j++];

}

while (i < L.size()) arr[k++] = L[i++];

while (j < R.size()) arr[k++] = R[j++];

return comps;

}

*long* *long* mergeSortRec(vector<*int*>& *arr*, *int* *l*, *int* *r*) {

*long* *long* comps = 0;

if (l < r) {

*int* m = l + (r - l) / 2;

comps += mergeSortRec(arr, l, m);

comps += mergeSortRec(arr, m + 1, r);

comps += merge(arr, l, m, r);

}

return comps;

}

*void* mergeSort(vector<*int*> *arr*) {

comparisons = mergeSortRec(arr, 0, arr.size() - 1);

}

*long* *long* quickSortRec(vector<*int*>& *arr*, *int* *low*, *int* *high*) {

*long* *long* comps = 0;

if (low < high) {

*int* pivot = arr[high], i = low - 1;

for (*int* j = low; j < high; j++) {

comps++;

if (arr[j] < pivot) {

i++;

swap(arr[i], arr[j]);

}

}

swap(arr[i + 1], arr[high]);

*int* pi = i + 1;

comps += quickSortRec(arr, low, pi - 1);

comps += quickSortRec(arr, pi + 1, high);

}

return comps;

}

*void* quickSort(vector<*int*> *arr*) {

comparisons = quickSortRec(arr, 0, arr.size() - 1);

}

*void* writeExcelHeader(*int* *row*, *int* *col\_offset*) {

worksheet\_write\_string(worksheet, row, col\_offset + 0, "n", header\_format);

worksheet\_write\_string(worksheet, row, col\_offset + 1, "Algorithm", header\_format);

worksheet\_write\_string(worksheet, row, col\_offset + 2, "Comparisons", header\_format);

worksheet\_write\_string(worksheet, row, col\_offset + 3, "Time(ms)", header\_format);

}

*void* writeExcelRow(*int* *row*, *int* *col\_offset*, *int* *n*, const string& *algo*, *long* *long* *comps*, *long* *double* *time\_ms*) {

worksheet\_write\_number(worksheet, row, col\_offset + 0, n, row\_format);

worksheet\_write\_string(worksheet, row, col\_offset + 1, algo.c\_str(), row\_format);

worksheet\_write\_number(worksheet, row, col\_offset + 2, comps, row\_format);

worksheet\_write\_number(worksheet, row, col\_offset + 3, time\_ms, row\_format);

}

*void* writeExcelSeparatorRow(*int* *row*, *int* *col\_offset*) {

for (*int* i = 0; i < 4; ++i)

worksheet\_write\_blank(worksheet, row, col\_offset + i, separator\_format);

}

*void* runSort(*void*(\**sortFunc*)(vector<*int*>), vector<*int*> *arr*, const string& *name*, *int* *n*, *int* *col\_offset*, vector<*long* *long*>& *comp\_list*, vector<*long* *double*>& *time\_list*) {

*auto* start = high\_resolution\_clock::now();

sortFunc(arr);

*auto* end = high\_resolution\_clock::now();

*long* *double* duration\_ns = duration\_cast<nanoseconds>(end - start).count();

*long* *double* duration\_ms = duration\_ns / 1'000'000.0;

cout << left << setw(12) << name << setw(16) << comparisons << fixed << setprecision(4) << duration\_ms << " ms" << endl;

if (generate\_excel) {

writeExcelRow(current\_row, col\_offset, n, name, comparisons, duration\_ms);

current\_row++;

}

comp\_list.push\_back(comparisons);

time\_list.push\_back(duration\_ms);

}

*void* runSort(*void*(\**sortFunc*)(vector<*int*>), vector<*int*> *arr*, const string& *name*, *int* *n*, *int* *col\_offset*, vector<*long* *long*>& *comp\_list*, vector<*long* *long*>& *time\_list*) {

*auto* start = high\_resolution\_clock::now();

sortFunc(arr);

*auto* end = high\_resolution\_clock::now();

*long* *long* duration\_ns = duration\_cast<nanoseconds>(end - start).count();

*long* *long* duration\_ms = duration\_ns / 1'000'000;

cout << left << setw(12) << name << setw(16) << comparisons << duration\_ms << " ms" << endl;

if (generate\_excel) {

writeExcelRow(current\_row, col\_offset, n, name, comparisons, duration\_ms);

current\_row++;

}

comp\_list.push\_back(comparisons);

time\_list.push\_back(duration\_ms);

}

*void* addSummaryAndChart(*int* *start\_row*) {

const *char*\* algo\_names[] = {"Selection", "Bubble", "Insertion", "MergeSort", "QuickSort"};

worksheet\_write\_string(worksheet, start\_row, 0, "n", header\_format);

for (*int* i = 0; i < 5; ++i)

worksheet\_write\_string(worksheet, start\_row, 1 + i, algo\_names[i], header\_format);

for (*size\_t* i = 0; i < input\_sizes.size(); ++i) {

worksheet\_write\_number(worksheet, start\_row + 1 + i, 0, input\_sizes[i], row\_format);

worksheet\_write\_number(worksheet, start\_row + 1 + i, 1, selection\_comparisons[i], row\_format);

worksheet\_write\_number(worksheet, start\_row + 1 + i, 2, bubble\_comparisons[i], row\_format);

worksheet\_write\_number(worksheet, start\_row + 1 + i, 3, insertion\_comparisons[i], row\_format);

worksheet\_write\_number(worksheet, start\_row + 1 + i, 4, merge\_comparisons[i], row\_format);

worksheet\_write\_number(worksheet, start\_row + 1 + i, 5, quick\_comparisons[i], row\_format);

}

lxw\_chart \*chart = workbook\_add\_chart(workbook, LXW\_CHART\_LINE);

chart\_axis\_set\_name(chart->x\_axis, "Input Size (n)");

chart\_axis\_set\_name(chart->y\_axis, "Element Comparisons");

chart\_title\_set\_name(chart, "Comparisons vs Input Size");

*int* chart\_start = start\_row + 1;

*int* chart\_end = chart\_start + input\_sizes.size() - 1;

for (*int* i = 0; i < 5; ++i) {

lxw\_chart\_series \*series = chart\_add\_series(chart, NULL, NULL);

chart\_series\_set\_name(series, algo\_names[i]);

chart\_series\_set\_categories(series, "Sheet1", chart\_start, 0, chart\_end, 0);

chart\_series\_set\_values(series, "Sheet1", chart\_start, 1 + i, chart\_end, 1 + i);

}

worksheet\_insert\_chart(worksheet, start\_row, 7, chart);

}

*void* addTimeChart(*int* *start\_row*) {

const *char*\* algo\_names[] = {"Selection", "Bubble", "Insertion", "MergeSort", "QuickSort"};

worksheet\_write\_string(worksheet, start\_row, 0, "n", header\_format);

for (*int* i = 0; i < 5; ++i)

worksheet\_write\_string(worksheet, start\_row, 1 + i, algo\_names[i], header\_format);

for (*size\_t* i = 0; i < input\_sizes.size(); ++i) {

worksheet\_write\_number(worksheet, start\_row + 1 + i, 0, input\_sizes[i], row\_format);

worksheet\_write\_number(worksheet, start\_row + 1 + i, 1, selection\_times[i], row\_format);

worksheet\_write\_number(worksheet, start\_row + 1 + i, 2, bubble\_times[i], row\_format);

worksheet\_write\_number(worksheet, start\_row + 1 + i, 3, insertion\_times[i], row\_format);

worksheet\_write\_number(worksheet, start\_row + 1 + i, 4, merge\_times[i], row\_format);

worksheet\_write\_number(worksheet, start\_row + 1 + i, 5, quick\_times[i], row\_format);

}

lxw\_chart \*chart = workbook\_add\_chart(workbook, LXW\_CHART\_LINE);

chart\_axis\_set\_name(chart->x\_axis, "Input Size (n)");

chart\_axis\_set\_name(chart->y\_axis, "Runtime (ms)");

chart\_title\_set\_name(chart, "Runtime vs Input Size");

*int* chart\_start = start\_row + 1;

*int* chart\_end = chart\_start + input\_sizes.size() - 1;

for (*int* i = 0; i < 5; ++i) {

lxw\_chart\_series \*series = chart\_add\_series(chart, NULL, NULL);

chart\_series\_set\_name(series, algo\_names[i]);

chart\_series\_set\_categories(series, "Sheet1", chart\_start, 0, chart\_end, 0);

chart\_series\_set\_values(series, "Sheet1", chart\_start, 1 + i, chart\_end, 1 + i);

}

worksheet\_insert\_chart(worksheet, start\_row, 7, chart);

}

*int* main() {

workbook = workbook\_new("projectexcel.xlsx");

worksheet = workbook\_add\_worksheet(workbook, NULL);

header\_format = workbook\_add\_format(workbook);

format\_set\_bold(header\_format);

format\_set\_font\_color(header\_format, LXW\_COLOR\_WHITE);

format\_set\_bg\_color(header\_format, 0x1F4E78);

format\_set\_align(header\_format, LXW\_ALIGN\_CENTER);

format\_set\_border(header\_format, LXW\_BORDER\_THIN);

format\_set\_border\_color(header\_format, LXW\_COLOR\_WHITE);

row\_format = workbook\_add\_format(workbook);

format\_set\_bg\_color(row\_format, 0xDDEEFF);

format\_set\_align(row\_format, LXW\_ALIGN\_CENTER);

format\_set\_border(row\_format, LXW\_BORDER\_THIN);

format\_set\_border\_color(row\_format, LXW\_COLOR\_WHITE);

separator\_format = workbook\_add\_format(workbook);

format\_set\_bg\_color(separator\_format, LXW\_COLOR\_WHITE);

srand(time(0));

vector<*int*> sizes = {100, 500, 1000, 1500, 2000, 3000, 4000, 5000};

for (*int* n : sizes) {

input\_sizes.push\_back(n);

if (block\_counter == 2) {

writeExcelSeparatorRow(current\_row, current\_col);

current\_row = 1;

current\_col += 5;

block\_counter = 0;

}

*long* *long* theo\_nlogn = static\_cast<*long* *long*>(n \* log2(n));

cout << "\nInput Size: " << n << endl;

cout << left << setw(12) << "Algorithm" << setw(16) << "Comparisons" << "Time" << endl;

cout << "--------------------------------------------------------" << endl;

cout << left << setw(12) << "nlogn" << setw(16) << theo\_nlogn << "-" << endl;

if (current\_row == 1)

writeExcelHeader(0, current\_col);

writeExcelRow(current\_row, current\_col, n, "nlogn", theo\_nlogn, 0);

current\_row++;

vector<*int*> base(n);

generate(base.begin(), base.end(), []() { return rand() % 101; });

runSort(selectionSort, base, "Selection", n, current\_col, selection\_comparisons, selection\_times);

runSort(bubbleSort, base, "Bubble", n, current\_col, bubble\_comparisons, bubble\_times);

runSort(insertionSort, base, "Insertion", n, current\_col, insertion\_comparisons, insertion\_times);

runSort(mergeSort, base, "MergeSort", n, current\_col, merge\_comparisons, merge\_times);

runSort(quickSort, base, "QuickSort", n, current\_col, quick\_comparisons, quick\_times);

block\_counter++;

if (n != sizes.back()) {

writeExcelSeparatorRow(current\_row, current\_col);

current\_row++;

}

}

cout << endl << endl;

*int* summary\_start\_row = current\_row + 3;

addSummaryAndChart(summary\_start\_row);

addTimeChart(summary\_start\_row + input\_sizes.size() + 5);

workbook\_close(workbook);

return 0;

}