Lab 14

Implementation of Sorting Techniques

The main objective of this lab is to **compare the efficiency of different sorting techniques** we have studied in the course.

**Bubble Sort**

**void** Bubble( **int** x[], **int** n)

{

**int** hold, j, pass;

**int** switched = TRUE;

**for** (pass = 0; pass < n-1 **&&** switched == TRUE; pass++)

{ // outer loop controls the number of passes switched = FALSE;

**for** ( j = 0; j < n-pass-1; j++)

{

// inner loop controls each individual pass

**if** (x[j] > x [j+1]) // elements out of order

{

switched = TRUE;

hold = x[j]; // interchange j and j+1 x[j] = x [j+1];

x[j+1] = hold;

} // end if

} // end inner for loop

} // end outer for loop

} // end Bubble

### **Insertion Sort**

##### Insertion sort (int x[], int n)

{

int j, k, y;

**/\* initially x[0] may be thought of as a sorted file of one element. After each repetition of the following loop, the elements x[0] through x[k] are in order \*/**

for (k=1; k<n; k++)

{

**/\* insert x[k] into sorted file \*/** y = x[k];

**/\* move down all elements greater than y \*/** for (j=k-1; j>=0 && y<x[j]; j++)

x[j+1] = x[j];

**/\* insert y at proper position \*/** x[j+1] = y;

}

}

### **Selection Sort**

##### void Selection sort(int x[], int n)

{

int key;

for(int a=0; a < n; a++) **// Outer loop for each pass**

{ key=a;

for(int b=a+1; b < n; b++) **// Inner loop to find the minimum**

{

if(x[b] < x[key])

key=b; **// Update index of the smallest element**

}

**// Swap the found minimum element with the first element of the unsorted part**

if (key > a)

{ int temp = x[a]; x[a] = x[key]; x[key] = temp;

}

}

}

### **Sorting using STL**

We will now start discussing some of the actual STL algorithms that can be applied to data structures. One of these is sorting.

**Example:**

Consider a simple example of a String class, and a vector of Strings:

#include <iostream>

#include <vector>

#include <algorithm>

#include <cstring> // for strdup

#include <cassert> // for assert

using namespace std;

class String {

char\* str;

public:

// Default constructor

String() : str(nullptr) {}

**// Constructor with string input**

String(const char\* s) {

str = strdup(s); // Duplicate the string

assert(str); // Assert that memory allocation succeeded

}

// Destructor to free allocated memory

~String() {

if (str) {

free(str); // Free dynamically allocated memory

}

}

// Less than operator to compare two String objects

bool operator<(const String& s) const {

return strcmp(str, s.str) < 0; // Compare strings lexicographically

}

// Conversion operator to char\* to output the string

operator const char\*() const {

return str;

}

};

int main() {

const char\* list[] = {"epsilon", "omega", "theta", "rho",

"alpha", "beta", "phi", "gamma", "delta"};

const int N = sizeof(list) / sizeof(list[0]);

// Create a vector of String objects

vector<String> v;

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

v.push\_back(String(list[i]));

}

// Shuffle the vector randomly

random\_device rd;

mt19937 g(rd()); // Mersenne Twister pseudo-random generator

shuffle(v.begin(), v.end(), g);

// Output the shuffled vector

cout << "Shuffled: ";

for (int j = 0; j < N; j++) {

cout << v[j] << " ";

}

cout << endl;

// Sort the vector

sort(v.begin(), v.end());

// Output the sorted vector

cout << "Sorted: ";

for (int j = 0; j < N; j++) {

cout << v[j] << " ";

}

cout << endl;

return 0;

}

##### **Output looks like:**

**phi delta beta theta omega alpha rho gamma epsilon alpha beta delta epsilon gamma omega phi rho theta**

**Computing Execution Time**

#include <iostream>

#include <ctime> // For time\_t, time(), and difftime

using namespace std;

int main() {

time\_t start, end; // Declare variables for time

char szInput[25]; // Array to hold user input

double dif; // Variable to hold the time difference

time(&start); // Record start time

cout << "Please, enter your name: ";

cin >> szInput; // Input the user's name (single word)

time(&end); // Record end time

dif = difftime(end, start); // Calculate the time difference

cout << "Hi " << szInput; // Output the greeting with the name

cout << " It took you " << dif << " seconds to type your name." << endl;

return 0; // Return 0 to indicate successful execution

}

**Output:**

Please, enter your name: John

Hi John It took you 0.002345 seconds to type your name

**Lab Tasks:**

Implement the following sorting techniques and count number of comparisons and exchanges.

1. **Bubble Sort**
2. **Selection Sort**
3. **Insertion Sort**
4. **Heap Sort**

* Run all the above techniques for the following values of N. Generate input data randomly and vary the input size N as below:

##### N = 10, 100, 1000, 5000, 10000, 20000, 50000

* Present your results in tabular and graphical (line graph) form. The tabulated output should have the following form:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **N** | **Bubble Sort** | | **Selection Sort** | | **Insertion Sort** | | **Heap Sort** | |
|  | **Comp.** | **Exch.** | **Comp.** | **Comp**  **.** | **Exch.** | **Exch.** | **Comp.** | **Exch.** |
| **10** |  |  |  |  |  |  |  |  |
| **100** |  |  |  |  |  |  |  |  |
| **1000** |  |  |  |  |  |  |  |  |
| **5000** |  |  |  |  |  |  |  |  |
| **10000** |  |  |  |  |  |  |  |  |
| **20000** |  |  |  |  |  |  |  |  |
| **50000** |  |  |  |  |  |  |  |  |