**Saad iqbal**

**Bee-6B**

**Reg# 32903**

**Task no# 01**

#include<iostream>//libraries

#include<time.h>

#include<vector>

#include<stdlib.h>

using namespace std;

void swap(int &i, int &j){//swap function

int temp;

temp = i;

i = j;

j = temp;

}

void print\_data(vector<int>a){//printing vector

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

cout << a[i] << " ";

}

}

vector<int> bubble\_sort(vector<int> a){//bubble\_sort algo

for (int i = a.size() - 1; i > 1; i--){

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

if (a[j]>a[j + 1])

swap(a[j], a[j + 1]);

}

}

return a;

}

vector<int> selection\_sort(vector<int> a){//selection\_sort algo

for (int i = a.size() - 1; i > 0; i--){

int temp = i;

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

if (a[j]>a[temp])

temp = j;

}

swap(a[temp], a[i]);

}

return a;

}

vector<int> insertion\_sort(vector<int> a){//insertion\_sort algo

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

int num = a[i];

int j = i - 1;

while (j >= 0 && a[j]>num){

a[j + 1] = a[j];

j = j - 1;

a[j + 1] = num;

}

}

return a;

}

vector<int> random(){//random number generation

vector<int>a;

for (int i = 0; i < 100; i++)

a.push\_back(rand() % 100 + 1);

return a;

}

int main(){//main

srand(time(NULL));

vector<int> x = random();//generation of data

cout << "My random data" << endl;

print\_data(x);//print data

cout << endl;

cout << "using bubble sort" << endl;

print\_data(bubble\_sort(x));//print sorted data using bubble sort

cout << endl;

cout << "using selection sort" << endl;

print\_data(selection\_sort(x));//print sorted data using selection sort

cout << endl;

cout << "using insertion sort" << endl;

print\_data(insertion\_sort(x));//print sorted data using insertion sort

cout << endl;

system("pause");

return 0;

}

**Task no# 02**

#include<iostream>//libraries

#include<time.h>

#include<vector>

#include<stdlib.h>

#include<fstream>

using namespace std;

void swap(int &i, int &j){//swap function

int temp;

temp = i;

i = j;

j = temp;

}

void print\_data(vector<int>a){//printing data

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

cout << a[i] << " ";

}

}

void bubble\_sort(vector<int> a){//bubble sorting

for (int i = a.size() - 1; i > 1; i--){

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

if (a[j]>a[j + 1])

swap(a[j], a[j + 1]);

}

}

}

void selection\_sort(vector<int> a){//selection sorting

for (int i = a.size() - 1; i > 0; i--){

int temp = i;

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

if (a[j]>a[temp])

temp = j;

}

swap(a[temp], a[i]);

}

}

void insertion\_sort(vector<int> a){//insertion sorting

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

int num = a[i];

int j = i - 1;

while (j >= 0 && a[j]>num){

a[j + 1] = a[j];

j = j - 1;

a[j + 1] = num;

}

}

}

vector<int> random(int y){//random number generation

vector<int>a;

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

a.push\_back(rand() % 100 + 1);

return a;

}

int main(){//main

int number = 100;

ofstream filed;//output file stream

filed.open("file.csv");

while (number <= 10000){//loop for analysing algo for more inputs

filed << number << "inputs" << endl;

int a, b;

srand(time(NULL));

vector<int> x = random(number);

cout << "using bubble sort" << endl;

a = clock();//clock function

bubble\_sort(x);

b = clock();

filed << "bubble sort" << "," << b - a << "msec" << endl;

cout << "using selection sort" << endl;

a = clock();

selection\_sort(x);

b = clock();

filed << "selection sort" << "," << b - a << "msec" << endl;

cout << "using insertion sort" << endl;

a = clock();

insertion\_sort(x);

b = clock();

filed << "insertion sort" << "," << b - a << "msec" << endl;

number \*= 10;//increment

}

filed.close();

system("pause");

return 0;

}

**Task no# 03**

#include<iostream>//libraries

#include<time.h>

#include<vector>

#include<stdlib.h>

#include<fstream>

#include<algorithm>

using namespace std;

void swap(int &i, int &j){//swap

int temp;

temp = i;

i = j;

j = temp;

}

void print\_data(vector<int>a){//printing data

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

cout << a[i] << " ";

}

}

void bubble\_sort(vector<int> a){//bubble sort

for (int i = a.size() - 1; i > 1; i--){

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

if (a[j]>a[j + 1])

swap(a[j], a[j + 1]);

}

}

}

void selection\_sort(vector<int> a){//selection sort

for (int i = a.size() - 1; i > 0; i--){

int temp = i;

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

if (a[j]>a[temp])

temp = j;

}

swap(a[temp], a[i]);

}

}

void insertion\_sort(vector<int> a){//insertion sort

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

int num = a[i];

int j = i - 1;

while (j >= 0 && a[j]>num){

a[j + 1] = a[j];

j = j - 1;

a[j + 1] = num;

}

}

}

vector<int> random(int y){//random

vector<int>a;

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

a.push\_back(rand() % 100 + 1);

return a;

}

int main(){//main

int number = 100;

ofstream filed;//output stream

filed.open("file.csv");

while (number <= 10000){

filed << "decending sorted" << number << "inputs" << endl;

int a, b;

srand(time(NULL));

vector<int> x = random(number);

// sort(x.begin(), x.end());

reverse(x.begin(), x.end());//reverse

cout << "using bubble sort" << endl;

a = clock();

bubble\_sort(x);

b = clock();

filed << "bubble sort" << "," << b - a << "msec" << endl;

cout << "using selection sort" << endl;

a = clock();

selection\_sort(x);

b = clock();

filed << "selection sort" << "," << b - a << "msec" << endl;

cout << "using insertion sort" << endl;

a = clock();

insertion\_sort(x);

b = clock();

filed << "insertion sort" << "," << b - a << "msec" << endl;

number \*= 10;//increment

}

filed.close();

system("pause");

return 0;

}

**Task#2 reasoning**

|  |  |
| --- | --- |
| 100inputs | Time |
| bubble sort | 0msec |
| selection sort | 0msec |
| insertion sort | 0msec |
| 1000inputs |  |
| bubble sort | 78msec |
| selection sort | 47msec |
| insertion sort | 46msec |
| 10000inputs |  |
| bubble sort | 7431msec |
| selection sort | 4453msec |
| insertion sort | 4438msec |

**Task2 running time analysis**

Result is nearly as same we imagining, for first 100 inputs, time is less than 1msec.

Bubble sort, check n^2 times.

Selection sort, pick number and put it into right place, somehow efficient.

Insertion sort, starts from one side and insert numbers at right place.

**Task#3 reasoning**

|  |  |
| --- | --- |
| ascending sorted100inputs | time |
| bubble sort | 0msec |
| selection sort | 0msec |
| insertion sort | 0msec |
| ascending sorted1000inputs |  |
| bubble sort | 47msec |
| selection sort | 47msec |
| insertion sort | 0msec |
| ascending sorted10000inputs |  |
| bubble sort | 4532msec |
| selection sort | 4471msec |
| insertion sort | 0msec |

**Task3 running time analysis for ascending data**

|  |  |
| --- | --- |
| decending sorted100inputs | time |
| bubble sort | 0msec |
| selection sort | 0msec |
| insertion sort | 0msec |
| decending sorted1000inputs |  |
| bubble sort | 109msec |
| selection sort | 47msec |
| insertion sort | 94msec |
| decending sorted10000inputs |  |
| bubble sort | 9992msec |
| selection sort | 4527msec |
| insertion sort | 8854msec |

**Task3 running time analysis for decending data**

**Best case:**

Bubble->half time than random

Selection->nearly same as bubble

Insertion->nearly takes no time

**Worst case:**

Bubble->much time input^2

Selection->time=(n^2)/2

Insertion->time=n^2