**Міністерство освіти і науки України**

**Національний університет «Запорізька політехніка»**

Кафедра програмних засобів

**ЗВІТ**

з лабораторної роботи №6

дисципліна: Алгоритмізація та програмування

тема: **«Программування с викроистанням алгоритмiв сортування»**

Варіант 22

Виконав:

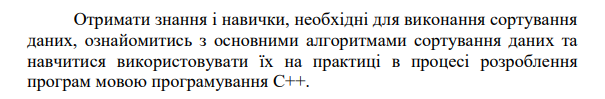
студентка гр. КНТ-210 Яценко А. А.

Перевірив:

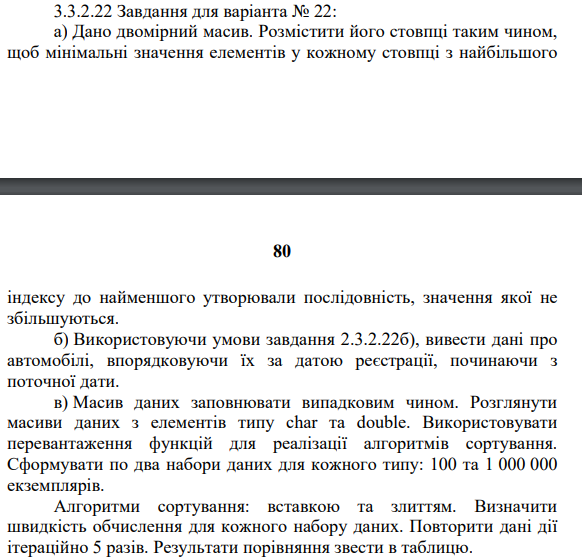
ст. викладач Качан О.І.

2020

**1.1 Мета роботи**

,

**1.2 Завдання:**



**1.4 Початкові коди програм**

а)

#include <iostream>

using namespace std;

/\* INPUT

2

5

1 5 4 2 3

1 5 4 2 3

\*/

int min(int\*\* matrix, int n, int column)

{

int result = matrix[0][column];

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

{

result = matrix[i][column] > result ? result : matrix[i][column];

}

return result;

}

void swap\_column(int\*\* matrix, int n, int c1, int c2)

{

int tmp;

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

{

tmp = matrix[i][c1];

matrix[i][c1] = matrix[i][c2];

matrix[i][c2] = tmp;

}

}

int\*\* sort\_matr(int\*\* matrix, int n, int m)

{

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

{

for (int j = i+1; j < m; j++)

{

//cout << min(matrix, n, i) << " " << min (matrix, n, j) << endl;

if (min(matrix, n, i) > min(matrix, n, j))

{

swap\_column(matrix, n, i, j);

}

}

}

return matrix;

}

int main()

{

int n,m;

cout << "n=";

cin >> n;

cout << "m=";

cin >> m;

int\*\* matrix = new int\*[n];

cout << "Matrix:" << endl;

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

matrix[i] = new int[m];

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

cin >> matrix[i][j];

}

}

int \*\*result = sort\_matr(matrix, n, m);

cout << "result:" << endl;

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

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

cout << result[i][j] << " ";

}

cout << endl;

}

}

б)

#include <iostream>

#include <map>

using namespace std;

struct Date

{

int day;

int month;

int year;

bool operator <(const Date& a) const

{

return

(

(a.year < year)

||

(a.year == year && a.month < month)

||

(a.year == year && a.month == month && a.day < day)

);

}

string tostring()

{

string result;

if (day < 10)

{

result += '0';

}

result += to\_string(day);

result += '-';

if (month < 10)

{

result += '0';

}

result += to\_string(month);

result += '-';

result += to\_string(year);

return result;

}

};

enum carType {hatchback = 'H', universal = 'U', sedan = 'S'};

struct Car

{

string model;

string manufacturer;

carType type;

int year;

Date registrationDate;

string FIO;

};

/\* INPUT

5

Kalina Lada S 2008 11-12-2020 Kulikov Valerii Maksimovich

Benz Mercedes H 2015 05-10-2019 Grigorenko Alexey Andreevich

Kalina Lada S 2008 11-12-2020 Sergeev Alexey Victorovich

Kalina Lada H 2008 11-01-2021 Kaleno Georgiy Valerievich

Car BMW U 1999 01-12-2020 Kulikov Valerii Maksimovich

\*/

void print\_car(Car car)

{

cout << car.model << " | " << car.manufacturer << " | " << (char)car.type << " | " << car.year << " | "

<< car.registrationDate.tostring() << " | " << car.FIO << endl;

}

void print\_stats(map<string, int> stats)

{

cout << "~~~~~~Stats~~~~~~" << endl;

for ( const auto &p : stats ) {

cout << p.first << " - " << p.second << "\n";

}

}

Date str\_to\_date(string str)

{

Date date = Date();

date.day = stoi(str.substr(0,2));

date.month = stoi(str.substr(3,2));

date.year = stoi(str.substr(6,4));

return date;

}

void sort\_cars\_by\_reg\_date(Car\* cars, int n)

{

Car tmp;

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

{

for (int j = i+1; j < n; j++)

{

if (cars[i].registrationDate < cars[j].registrationDate)

{

tmp = cars[i];

cars[i] = cars[j];

cars[j] = tmp;

}

}

}

}

int main()

{

int n;

cout << "Cars count = ";

cin >> n;

cout << "Enter players in below format:" << endl;

cout << "Model | Manufacturer | Type(H-hatchback, U-universal, S-sedan | Year | Registration Date | FIO" << endl;

cout << "0. Kalina Lada S 2008 11-12-2020 Kulikov Valerii Maksimovich" << endl;

Car\* cars = new Car[n];

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

{

cout << i+1 << ". ";

cin >> cars[i].model;

cin >> cars[i].manufacturer;

char type;

cin >> type;

switch (tolower(type))

{

case 'h':

cars[i].type = hatchback;

break;

case 'u':

cars[i].type = universal;

break;

case 's':

cars[i].type = sedan;

break;

}

cin >> cars[i].year;

string reg\_date;

cin >> reg\_date;

cars[i].registrationDate = str\_to\_date(reg\_date);

string F, I, O;

cin >> F >> I >> O;

cars[i].FIO = F + " " + I + " " + O;

}

sort\_cars\_by\_reg\_date(cars, n);

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

{

print\_car(cars[i]);

}

}

в)

#include <iostream>

#include <cstdlib>

#include <chrono>

using namespace std;

char\* copy(char\* arr, int n)

{

char\* result = new char[n];

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

{

result[i] = arr[i];

}

return result;

}

double\* copy(double\* arr, int n)

{

double\* result = new double[n];

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

{

result[i] = arr[i];

}

return result;

}

void Merge(double\* arr,int p, int q,int r)

{

int n1,n2,i,j,k;

n1=q-p+1;

n2=r-q;

double\* L = new double[n1];

double\* R = new double[n2];

for(i=0;i<n1;i++)

{

L[i]=arr[p+i];

}

for(j=0;j<n2;j++)

{

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

}

i=0,j=0;

for(k=p;i<n1&&j<n2;k++)

{

if(L[i]<R[j])

{

arr[k]=L[i++];

}

else

{

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

}

}

while(i<n1)

{

arr[k++]=L[i++];

}

while(j<n2)

{

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

}

}

void MergeSort(double\* arr,int p,int r)

{

int q;

if(p<r)

{

q=(p+r)/2;

MergeSort(arr,p,q);

MergeSort(arr,q+1,r);

Merge(arr,p,q,r);

}

}

void Merge(char\* arr,int p, int q,int r)

{

int n1,n2,i,j,k;

n1=q-p+1;

n2=r-q;

char\* L = new char[n1];

char\* R = new char[n2];

for(i=0;i<n1;i++)

{

L[i]=arr[p+i];

}

for(j=0;j<n2;j++)

{

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

}

i=0,j=0;

for(k=p;i<n1&&j<n2;k++)

{

if(L[i]<R[j])

{

arr[k]=L[i++];

}

else

{

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

}

}

while(i<n1)

{

arr[k++]=L[i++];

}

while(j<n2)

{

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

}

}

void MergeSort(char\* arr,int p,int r)

{

int q;

if(p<r)

{

q=(p+r)/2;

MergeSort(arr,p,q);

MergeSort(arr,q+1,r);

Merge(arr,p,q,r);

}

}

void insert\_sort(double\* arr, int n)

{

double key;

int i, j;

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

{

key = arr[i];

j = i - 1;

while (j >= 0 && arr[j] > key)

{

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

j = j - 1;

}

arr[j + 1] = key;

}

}

void insert\_sort(char\* arr, int n)

{

char key;

int i, j;

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

{

key = arr[i];

j = i - 1;

while (j >= 0 && arr[j] > key)

{

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

j = j - 1;

}

arr[j + 1] = key;

}

}

void fill\_data(char\* arr, int count)

{

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

{

arr[i] = 'A' + rand()%24;

}

}

void fill\_data(double\* arr, int count)

{

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

{

arr[i] = rand()/0.98789;

}

}

int main()

{

srand(time(NULL));

char\* data\_char\_1 = new char[100];

char\* data\_char\_2 = new char[1000000];

double\* data\_double\_1 = new double[100];

double\* data\_double\_2 = new double[1000000];

fill\_data(data\_char\_1, 100);

fill\_data(data\_char\_2, 1000000);

fill\_data(data\_double\_1, 100);

fill\_data(data\_double\_2, 1000000);

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

{

cout << "~~~~~~~~~TEST " << i+1 << "~~~~~~~~~" << endl;

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

insert\_sort(copy(data\_char\_1, 100), 100);

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

double time\_taken =

chrono::duration\_cast<chrono::nanoseconds>(end - start).count();

time\_taken \*= 1e-9;

cout << "Time taken by insert sort CHAR on 100 elements " << fixed

<< time\_taken;

cout << " sec" << endl;

start = chrono::high\_resolution\_clock::now();

insert\_sort(copy(data\_double\_1, 100), 100);

end = chrono::high\_resolution\_clock::now();

time\_taken =

chrono::duration\_cast<chrono::nanoseconds>(end - start).count();

time\_taken \*= 1e-9;

cout << "Time taken by insert sort DOUBLE on 100 elements " << fixed

<< time\_taken;

cout << " sec" << endl;

start = chrono::high\_resolution\_clock::now();

insert\_sort(copy(data\_char\_2, 1000000), 1000000);

end = chrono::high\_resolution\_clock::now();

time\_taken =

chrono::duration\_cast<chrono::nanoseconds>(end - start).count();

time\_taken \*= 1e-9;

cout << "Time taken by insert sort CHAR on 1000000 elements " << fixed

<< time\_taken;

cout << " sec" << endl;

start = chrono::high\_resolution\_clock::now();

insert\_sort(copy(data\_double\_2, 1000000), 1000000);

end = chrono::high\_resolution\_clock::now();

time\_taken =

chrono::duration\_cast<chrono::nanoseconds>(end - start).count();

time\_taken \*= 1e-9;

cout << "Time taken by insert sort DOUBLE on 1000000 elements " << fixed

<< time\_taken;

cout << " sec" << endl;

start = chrono::high\_resolution\_clock::now();

MergeSort(copy(data\_char\_1, 100), 0, 100);

end = chrono::high\_resolution\_clock::now();

time\_taken =

chrono::duration\_cast<chrono::nanoseconds>(end - start).count();

time\_taken \*= 1e-9;

cout << "Time taken by merge sort CHAR on 100 elements " << fixed

<< time\_taken;

cout << " sec" << endl;

start = chrono::high\_resolution\_clock::now();

MergeSort(copy(data\_double\_1, 100), 0, 100);

end = chrono::high\_resolution\_clock::now();

time\_taken =

chrono::duration\_cast<chrono::nanoseconds>(end - start).count();

time\_taken \*= 1e-9;

cout << "Time taken by merge sort DOUBLE on 100 elements " << fixed

<< time\_taken;

cout << " sec" << endl;

start = chrono::high\_resolution\_clock::now();

MergeSort(copy(data\_char\_2, 1000000), 0, 1000000);

end = chrono::high\_resolution\_clock::now();

time\_taken =

chrono::duration\_cast<chrono::nanoseconds>(end - start).count();

time\_taken \*= 1e-9;

cout << "Time taken by merge sort CHAR on 1000000 elements " << fixed

<< time\_taken;

cout << " sec" << endl;

start = chrono::high\_resolution\_clock::now();

MergeSort(copy(data\_double\_2, 1000000), 0, 1000000);

end = chrono::high\_resolution\_clock::now();

time\_taken =

chrono::duration\_cast<chrono::nanoseconds>(end - start).count();

time\_taken \*= 1e-9;

cout << "Time taken by merge sort DOUBLE on 1000000 elements " << fixed

<< time\_taken;

cout << " sec" << endl;

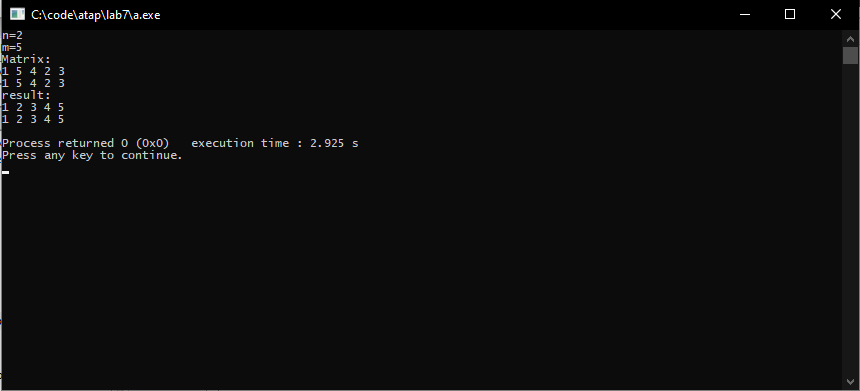
}

}

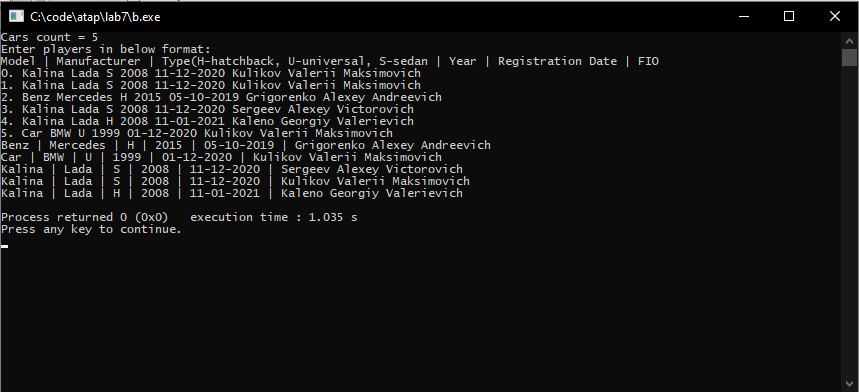
**1.5 Результат роботи програм**

Скриншоти

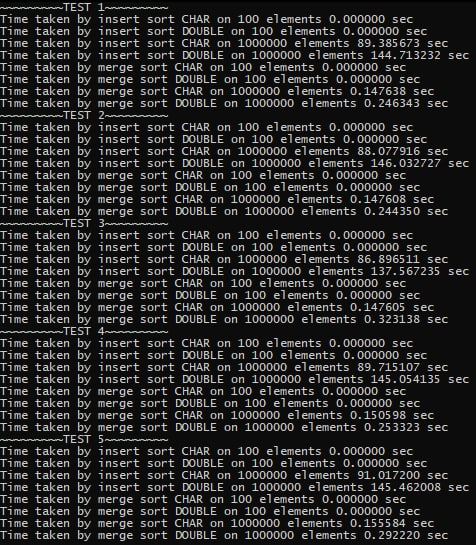
а)



б)



в)

ё

**1.7 Висновки**

Отримав знання і навички, необхідні для програмування з викроистанням алгоритмiв сортування, та навчився використовувати їх на практиці в процесі розроблення програм мовою програмування С++.