**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра информационных систем**

отчет

**по курсовой работе**

**по дисциплине «Программирование»**

Тема: Объединение нескольких работ

|  |  |  |
| --- | --- | --- |
| Студент гр. 2372 |  | Филимонов Е.П. |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2022

**Цель работы.**

Необходимо объединить все 4 лабораторные работы в единый проект. Нужно добавить инфраструктуру переключения между заданиями (интерактивное меню).

**Постановка задачи.**

Объединим все четыре работы и добавим интерактивное меню.

**Основные теоретические положения.**

Знания, которые мы приобрели за 4 практические работы.

**Выполнение работы.**

Мы добавляем структуру switch и вставляем в них каждую нашу программу.

**Выводы.**

Данная работа выполняется довольно легко, с помощью структуры switch case.

Приложение А

Полный код программы

#include <iostream>

#include <stdlib.h>

#include <iomanip>

#include <chrono>

#include <clocale>

#include <windows.h>

#include <ctime>

#include <string>

#include <fstream>

using namespace std;

void swap(int\* a, int\* b)

{

int temp;

temp = \*a;

\*a = \*b;

\*b = temp;

}

string Deletespace(string s) {

for (int i = 0; i < s.length(); i++) {

if ((s[i] == ' ') && (s[i + 1] == ' ')) {

while (s[i + 1] == ' ')

s.erase(i + 1, 1);

i++;

}

}

return s;

}

void linearfnd(string s, string l)

{

int p = 0;

if (s.find(l) == string::npos)

{

cout << "Dont have this(" << endl;

}

else

{

while ((p = s.find(l, p)) != string::npos)

{

cout << p << " ";

p += l.length();

}

}

cout << endl;

}

string DeleteZn(string s) {

for (int i = 0; i < s.length(); i++) {

{

if ((s[i] == '.') && (s[i + 1] == '.')) {

while (s[i + 1] == '.')

s.erase(i + 1, 1);

i++;

}

if ((s[i] == ',') && (s[i + 1] == ',')) {

while (s[i + 1] == ',')

s.erase(i + 1, 1);

i++;

}

if ((s[i] == ';') && (s[i + 1] == ';')) {

while (s[i + 1] == ';')

s.erase(i + 1, 1);

i++;

}

if ((s[i] == ':') && (s[i + 1] == ':')) {

while (s[i + 1] == ':')

s.erase(i + 1, 1);

i++;

}

if ((s[i] == '?') && (s[i + 1] == '?')) {

while (s[i + 1] == '?')

s.erase(i + 1, 1);

i++;

}

if ((s[i] == '!') && (s[i + 1] == '!')) {

while (s[i + 1] == '!')

s.erase(i + 1, 1);

i++;

}

}

}

return s;

}

string regEdit(string s)

{

for (int i = 1; i < s.length() - 1; i++)

{

if (s[i - 1] == ' ')

{

i++;

}

s[i] = tolower(s[i]);

}

return s;

}

void Double(string s)

{

int a = 0;

bool ds = false;

for (int i = 1; i < s.length(); i++)

{

if (tolower(s[i]) == tolower(s[i + 1]))

{

ds = true;

}

if (s[i] == ' ')

{

if (ds == true) {

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

{

if (s[j] != '.' && s[j] != ',' && s[j] != '!' && s[j] != '?' && s[j] != ';' && s[j] != ':')

{

cout << s[j];

}

}

cout << " ";

a = i + 1;

ds = false;

}

else

{

a = i + 1;

ds = false;

}

}

}

}

string PropisBuk(string s)

{

s[0] = toupper(s[0]);

for (int i = 1; i < s.length() - 1; i++)

{

if (s[i - 1] == ' ')

{

s[i] = toupper(s[i]);

}

}

return s;

}

void InsertSort(int a[])

{

for (int i = 1; i < 100; i++)

{

int k = i - 1;

while (k >= 0 && a[k] > a[k + 1])

{

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

k--;

}

}

}

void Bubble(int a[])

{

for (int i = 0; i < 100 - 1; i++)

{

for (int j = 0; j < 100 - 1 - i; j++)

{

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

{

int vr = 0;

vr = a[j];

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

a[j + 1] = vr;

}

}

}

}

void ShakerSort(int a[])

{

int i, j, k;

int n = 100;

int f = 0;

for (i = f; i < 100; i++)

{

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

{

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

{

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

}

}

for (k = n - 1; k >= f; k--)

{

if (a[k] < a[k - 1])

{

swap(&a[j], &a[j - 1]);

}

}

}

}

void QuickSort(int\* arr, int end, int begin)

{

int mid;

int f = begin;

int l = end;

mid = arr[(f + l) / 2];

while (f < l)

{

while (arr[f] < mid) f++;

while (arr[l] > mid) l--;

if (f <= l)

{

swap(arr[f], arr[l]);

f++;

l--;

}

}

if (begin < l) QuickSort(arr, l, begin);

if (f < end) QuickSort(arr, end, f);

}

void PrintArray(int\* arr, int n) {

int\* p = arr;

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

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

cout << \*p << " ";

p++;

}

cout << endl;

}

}

int main() {

bool d=true;

while (d==true) {

int task;

cout << "Select task:\n"

"1. First practical\n"

"2. Second practical\n"

"3. Third practical\n"

"4. Fourth practical\n";

cin >> task;

if (task == 0) {

d = false;

break;

}

cout << '\n' << '\n';

switch (task)

{

case 1:

{

while (true) {

int task;

cout << "Select task:\n"

"1. Memory in bytes\n"

"2. Binary representation of a number(int) in memory\n"

"3. Binary representation of a number(float) in memory\n"

"0. Go back\n";

cin >> task;

cout << '\n' << '\n';

switch (task) {

case 1: {

cout << sizeof(int) << " - Size of int" << "\n" << sizeof(short int) << " - Size of short int" << "\n" << sizeof(long int) << " - Size of long int" << "\n"

<< sizeof(float) << " - Size of float" << "\n" << sizeof(double) << " - Size of double" << "\n" << sizeof(long double) << " - Size of long double" << "\n" << sizeof(char) << " - Size of char" << "\n" << sizeof(bool) << " - Size of bool" << "\n" << endl;

break;

}

case 2: {

int a;

cout << "Enter number(int):";

cin >> a;

cout << "Binary representation of a number(int) in memory:" << '\n';

unsigned order = sizeof(int) \* 8, mask = pow(2, 31);

for (int i = 1; i <= order; i++)

{

putchar(a & mask ? '1' : '0');

mask >>= 1;

if (i % 8 == 0)

{

cout << " ";

}

if (i % order - 1 == 0)

{

cout << " ";

}

}

cout << '\n' << endl;

break;

}

case 3: {

union {

int b;

float c;

};

unsigned order1 = sizeof(int) \* 8, mask1 = pow(2, 31);

cout << "Enter number(float):";

cin >> c;

cout << "Binary representation of a number(float) in memory:" << '\n';

for (int j = 1; j <= order1; j++)

{

putchar(b & mask1 ? '1' : '0');

mask1 >>= 1;

if (j % 8 == 0)

{

cout << " ";

}

if (j % order1 - 1 == 0)

{

cout << " ";

}

}

cout << "\n" << endl;

break;

}

case 0: {

d = false;

}

default:

{

break;

}

}

}

}

case 2:

{

srand(time(NULL));

int m[100];

int f[100];

int z;

int MaxA = -100;

int MinA = 100;

while (true) {

int task;

cout << "Select task:\n"

"1. Create array\n"

"2. Sort array\n"

"3. Minimum and Maximum\n"

"4. Avarage of Minimum and Maximum\n"

"5. The number of elements, are less \"a\"\n"

"6. The number of elements, are more \"b\"\n"

"8. Element replacement \"a\" & \"b\"\n"

"9. Close programm\n";

cin >> task;

cout << '\n' << '\n';

switch (task)

{

case 1:

{

for (z = 0; z < 100; z++)

{

int rando = (rand() % 100);

int zn = (rand() % 2);

if (zn == 0)

{

rando = -rando;

}

m[z] = rando;

f[z] = m[z];

}

cout << "Unsorted array" << endl;

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

cout << setw(8) << left << m[i];

cout << endl << '\n' << '\n';

break;

}

case 2:

{

int sorttype;

cout << "Choice sort:\n" <<

" 1. Bubble Sort\n" <<

" 2. Shaker Sort\n" <<

" 3. Insertion Sort\n" <<

" 4. Quick Sort\n";

cin >> sorttype;

switch (sorttype)

{

case 1: {

auto start = chrono::steady\_clock::now();

Bubble(m);

auto end = chrono::steady\_clock::now();

cout << "Sorted array" << endl;

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

cout << setw(8) << left << m[i];

cout << endl;

cout << "Elapsed time in nanoseconds: "

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

<< " ns " << endl;

cout << endl << '\n' << '\n';

break;

}

case 2: {

auto start = chrono::steady\_clock::now();

ShakerSort(m);

auto end = chrono::steady\_clock::now();

cout << "Sorted array" << endl;

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

cout << setw(8) << left << m[i];

cout << endl;

cout << "Elapsed time in nanoseconds: "

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

<< " ns " << endl;

cout << endl << '\n' << '\n';

break;

}

case 3: {

auto start = chrono::steady\_clock::now();

InsertSort(m);

auto end = chrono::steady\_clock::now();

cout << "Sorted array" << endl;

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

cout << setw(8) << left << m[i];

cout << endl;

cout << "Elapsed time in nanoseconds: "

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

<< " ns " << endl;

cout << endl << '\n' << '\n';

break;

}

case 4: {

auto start = chrono::steady\_clock::now();

QuickSort(m, 99, 0);

auto end = chrono::steady\_clock::now();

cout << "Sorted array" << endl;

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

cout << setw(8) << left << m[i];

cout << endl;

cout << "Elapsed time in nanoseconds: "

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

<< " ns " << endl;

cout << endl << '\n' << '\n';

break;

}

default: {

break;

}

}

break;

}

case 3: {

InsertSort(m);

auto start1 = chrono::steady\_clock::now();

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

{

if (f[i] > MaxA)

MaxA = f[i];

if (f[i] < MinA)

MinA = f[i];

}

auto end1 = chrono::steady\_clock::now();

cout << "In sorted array" << endl;

cout << "Max:" << MaxA << " " << "Min:" << MinA;

cout << endl;

cout << "Elapsed time in nanoseconds: "

<< chrono::duration\_cast<chrono::nanoseconds>(end1 - start1).count()

<< " ns" << endl;

cout << endl;

cout << "In unsorted array" << endl;

auto start2 = chrono::steady\_clock::now();

cout << "Max:" << m[99] << " " << "Min:" << m[0];

auto end2 = chrono::steady\_clock::now();

cout << endl;

cout << "Elapsed time in nanoseconds: "

<< chrono::duration\_cast<chrono::nanoseconds>(end2 - start2).count()

<< " ns " << endl;

cout << endl << '\n';

break;

}

case 4: {

int s = 0;

float g = (MaxA + MinA) / 2;

cout << "Avarage of Max and Min in unsorted array:" << g;

cout << endl;

cout << "Element indices: ";

auto start3 = chrono::steady\_clock::now();

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

{

if (g == f[i])

{

cout << i << " ";

s += 1;

}

}

if (s == 0)

{

cout << "no element";

}

auto end3 = chrono::steady\_clock::now();

cout << '\n' << "Amount of element:" << s;

cout << endl;

cout << "Elapsed time in nanoseconds: "

<< chrono::duration\_cast<chrono::nanoseconds>(end3 - start3).count()

<< " ns " << endl;

cout << endl;

int l = 0;

float q = (m[99] + m[0]) / 2;

cout << "Avarage of Max and Min in sorted array:" << q;

cout << endl;

cout << "Element indices: ";

auto start4 = chrono::steady\_clock::now();

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

{

if (q == m[i])

{

cout << i << " ";

l += 1;

}

}

if (l == 0)

{

cout << "no element";

}

auto end4 = chrono::steady\_clock::now();

cout << '\n' << "Amount of element:" << l;

cout << endl;

cout << "Elapsed time in nanoseconds: "

<< chrono::duration\_cast<chrono::nanoseconds>(end4 - start4).count()

<< " ns" << endl;

cout << endl;

cout << endl;

break;

}

case 5: {

Bubble(m);

int e;

int y = 0;

cout << "Enter the number:";

cin >> e;

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

{

if (e > m[i])

{

y += 1;

}

}

cout << "Amount of elements:" << y;

cout << endl;

cout << endl;

break;

}

case 6: {

Bubble(m);

int n;

int p = 0;

cout << "Enter the number:";

cin >> n;

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

{

if (n < m[i])

{

p += 1;

}

}

cout << "Amount of elements:" << p;

cout << endl;

cout << endl;

break;

}

case 8: {

int a, b;

Bubble(f);

cout << "Enter 2 indices(0-99): ";

cin >> a >> b;

auto start5 = chrono::steady\_clock::now();

swap(f[a], f[b]);

auto end5 = chrono::steady\_clock::now();

cout << "Array with swapped elements" << endl;

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

cout << setw(8) << left << f[i];

}

cout << endl;

cout << '\n';

cout << "Elapsed time in nanoseconds: "

<< chrono::duration\_cast<chrono::nanoseconds>(end5 - start5).count()

<< " ns" << endl;

cout << '\n';

break;

}

case 9: {exit(0);

break;

}

default:

{

break;

}

}

}

break;

}

case 3:

{

srand(time(0));

HANDLE hStdout;

COORD destCoord;

hStdout = GetStdHandle(STD\_OUTPUT\_HANDLE);

const int n = 6;

int arr[n][n];

int\* p = \*arr;;

int k = 1;

int move = 1;

int j = 0;

int i = 0;

int task;

char variant;

int temp[n][n];

int t = 0;

int count = 0;

char action;

int number;

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

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

arr[i][j] = 0;

}

}

while (true) {

int task;

cout << "Select task:\n"

"1. Create a matrix\n"

"2. Move matrix blocks\n"

"3. Sorted matrix\n"

"4. Actions with the matrix\n";

cin >> task;

cout << '\n' << '\n';

switch (task) {

case 1:

cout << "a or b?" << endl;

cin >> variant;

system("cls");

switch (variant) {

case 'a':

k = 1;

i = 0;

j = 0;

move = 1;

p = \*arr;

destCoord.X = j;

destCoord.Y = i;

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

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

arr[i][j] = 0;

}

}

while (k <= n \* n) {

if (k == n) {

move = n;

}

if (\*(p + move) == 0 || k == n \* n) {

\*p = rand() % (n \* n) + 1;

destCoord.X = 5 \* j;

destCoord.Y = 3 \* i;

SetConsoleCursorPosition(hStdout, destCoord);

cout << \*p << '\r';

cout.flush();

Sleep(100);

if (move == 1) {

j++;

}

else if (move == n) {

i++;

}

else if (move == -1) {

j--;

}

else if (move == -n) {

i--;

}

p += move;

k++;

}

else {

switch (move) {

case 1:

move = n;

break;

case n:

move = -1;

break;

case -1:

move = -n;

break;

case -n:

move = 1;

break;

default:

break;

}

}

}

cout << "\n\n\n\n\n\n\n" << endl;

break;

case 'b':

move = n;

k = 1;

i = 0;

j = 0;

p = \*arr;

destCoord.X = j;

destCoord.Y = i;

while (k <= n \* n) {

\*p = rand() % (n \* n) + 1;

destCoord.X = 5 \* j;

destCoord.Y = 3 \* i;

SetConsoleCursorPosition(hStdout, destCoord);

cout << \*p << '\r';

Sleep(100);

if (move == n && k % n != 0) {

i++;

p += move;

}

else if (move == -n && k % n != 0) {

i--;

p += move;

}

else {

move = -move;

j++;

p += 1;

}

k++;

}

cout << "\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n" << endl;

break;

default:

break;

}

break;

case 2:

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

for (j = 0; j < n; j++) {

temp[i][j] = arr[i][j];

}

}

cout << "Array:\n";

PrintArray(\*arr, n);

cout << '\n';

cout << "Choose variant (a,b,c,d)\n";

cin >> variant;

switch (variant) {

case 'a':

for (i = 0; i < n / 2; i++) {

for (j = 0; j < n / 2; j++) {

arr[i][j + n / 2] = temp[i][j];

arr[i][j] = temp[i + n / 2][j];

arr[i + n / 2][j + n / 2] = temp[i][j + n / 2];

arr[i + n / 2][j] = temp[i + n / 2][j + n / 2];

}

}

break;

case 'b':

for (i = 0; i < n / 2; i++) {

for (j = 0; j < n / 2; j++) {

arr[i][j] = temp[i + n / 2][j + n / 2];

arr[i][j + n / 2] = temp[i + n / 2][j];

arr[i + n / 2][j] = temp[i][j + n / 2];

arr[i + n / 2][j + n / 2] = temp[i][j];

}

}

break;

case 'c':

for (i = 0; i < n / 2; i++) {

for (j = 0; j < n / 2; j++) {

arr[i][j] = temp[i + n / 2][j];

arr[i][j + n / 2] = temp[i + n / 2][j + n / 2];

arr[i + n / 2][j] = temp[i][j];

arr[i + n / 2][j + n / 2] = temp[i][j + n / 2];

}

}

break;

case 'd':

for (i = 0; i < n / 2; i++) {

for (j = 0; j < n / 2; j++) {

arr[i][j] = temp[i][j + n / 2];

arr[i][j + n / 2] = temp[i][j];

arr[i + n / 2][j] = temp[i + n / 2][j + n / 2];

arr[i + n / 2][j + n / 2] = temp[i + n / 2][j];

}

}

break;

}

cout << '\n';

PrintArray(\*arr, n);

cout << '\n';

break;

case 3:

cout << "Array:\n";

PrintArray(\*arr, n);

cout << '\n';

k = 0;

count = 0;

p = \*arr;

while (k < n \* n) {

while (count < n \* n - 1) {

if (\*p > \*(p + 1)) {

t = \*p;

\*p = \*(p + 1);

\*(p + 1) = t;

}

count++;

p++;

}

k++;

p = \*arr;

count = 0;

}

cout << "Sorted array:\n";

PrintArray(\*arr, n);

cout << '\n';

break;

case 4:

k = 0;

p = \*arr;

cout << "Array:\n";

PrintArray(\*arr, n);

cout << '\n';

cout << "Choose action (+, -, \*, /)\n";

cin >> action;

cout << "Choose number:\n";

cin >> number;

cout << '\n';

switch (action) {

case '+':

while (k < n \* n) {

\*p = \*p + number;

k++;

p++;

}

case '-':

while (k < n \* n) {

\*p = \*p - number;

k++;

p++;

}

case '\*':

while (k < n \* n) {

\*p = \*p \* number;

k++;

p++;

}

case '/':

while (k < n \* n) {

if (number == 0) {

break;

}

\*p = \*p / number;

k++;

p++;

}

default: break;

}

PrintArray(\*arr, n);

cout << '\n';

default:

break;

}

}

}

case 4:

{

string s;

while (true) {

string task;

cout << "Select task:\n"

"1. Enter text\n"

"2. Fixed text\n"

"3. Identical letters\n"

"4. Uppercase letter\n"

"5. Substrings\n";

getline(cin, task);

cout << '\n' << '\n';

if (task == "1") {

string task2;

cout << "Select way:\n"

"1. Keyboard\n"

"2. File\n";

getline(cin, task2);

if (task2 == "1") {

cout << "Enter text:" << '\n';

getline(cin, s);

s = s + " ";

cout << '\n';

cout << "Your text:" << '\n';

cout << s << endl;

cout << '\n';

}

if (task2 == "2") {

ifstream file("C://123.txt");

char a;

while (!file.eof())

{

file.get(a);

s.push\_back(a);

}

cout << "Your text:" << '\n';

cout << s << '\n';

}

}

if (task == "2") {

s = DeleteZn(s);

s = Deletespace(s);

s = regEdit(s);

s = DeleteZn(s);

cout << "Corect text:" << '\n';

cout << s << endl;

cout << '\n';

}

if (task == "3") {

cout << "Words with the same letters: ";

Double(s);

cout << endl;

}

if (task == "4") {

cout << "Text, in which all the words with a capital letter:" << '\n';

s = PropisBuk(s);

cout << s << endl;

}

if (task == "5") {

string l;

cout << "Enter word: ";

cin >> l;

linearfnd(s, l);

}

}

}

}

}

}