Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра систем штучного інтелекту



**Звіт**

про виконання

**Лабораторних та практичних робіт № 6,8,9**

***з дисципліни:*** «Мови та парадигми програмування»

***з розділу***: «Робота з файлами, системи числення, бібліотеки»

***Виконав(ла):***

студент групи ШІ-14

Грицина Ярослав

# **Тема роботи:**

# Робота з файлами, системи числення, бібліотеки

# **Мета роботи:**

Навчитись працювати з файлами, системами числення, бібліотеками.

# **Теоретичні відомості:**

1. Теоретичні відомості з переліком важливих тем:

* Тема №1: Файли.
* Тема №2: Системи числення.
* Тема №3: Бібліотеки.

1. Індивідуальний план опрацювання теорії:

* Тема №1: Файли.
  + Джерела Інформації
    - Стаття https://www.w3schools.in/cplusplus/working-with-files.
  + Що опрацьовано:
    - Відкривання і закривання файлів
    - Ввід і вивід даних
  + Статус: Ознайомлений
  + Початок опрацювання теми: 09.12
  + Звершення опрацювання теми: 12.12
* Тема №2: Системи числення.
  + Джерела Інформації:
    - Стаття https://uk.wikipedia.org/wiki/%D0%9F%D0%BE%D0%B7%D0%B8%D1%86%D1%96%D0%B9%D0%BD%D0%B0\_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0\_%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%BD%D1%8F.
  + Що опрацьовано:
    - Двійкову, вісімкову і шістнадцяткову систему числення
    - Перехід з однієї системи в іншу
  + Статус: Ознайомлений
  + Початок опрацювання теми: 09.12
  + Звершення опрацювання теми: 12.12
* Тема №3: Бібліотеки.
  + Джерела Інформації:
    - Стаття https://www.geeksforgeeks.org/the-c-standard-template-library-stl/.
  + Що опрацьовано:
    - Стандартний набір бібліотек STL.
  + Статус: Ознайомлений
  + Початок опрацювання теми: 09.12
  + Звершення опрацювання теми: 12.12

# **Виконання роботи:**

## **1. Опрацювання завдання та вимог до програм та середовища:**

Завдання №1 Lab# programming: VNS Lab 6

* 12
* Виконати ввід рядка, використовуючи функцію gets(s) і здійснити обробку рядка

Завдання №2 Lab# programming: VNS Lab 8

* 12
* Сформувати двійковий файл, виконати формування, друк, додавання й знищення

елементів оформити у вигляді функцій.

Завдання №3 Lab# programming: VNS Lab 9

* 12
* Створити текстовий файл F1 не менше, ніж з 10 рядків і записати в нього

інформацію, виконати завдання.

Завдання №4 Lab# programming: Algotester Lab 4

* 3
* Потрібно виконати 2 варіанти програми

Завдання №5 Lab# programming: Algotester Lab 6

* 3

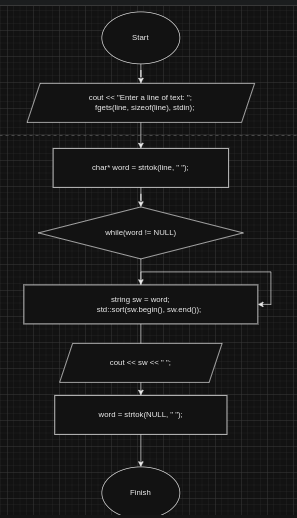
Завдання №6 Lab# programming: Class Practice 1

Завдання №7 Lab# programming: Class Practice 2

## **2. Дизайн та планована оцінка часу виконання завдань:**

Програма №1 Lab# programming: VNS Lab 6

* Блок-схема



* Планований час на реалізацію 2 год

Програма №2 Lab# programming: VNS Lab 8

* Планований час на реалізацію 2 год

Програма №3 Lab# programming: VNS Lab 9

* Планований час на реалізацію 2 год

Програма №4 Lab# programming: Algotester Lab 4

* Планований час на реалізацію 2 год

Програма №5 Lab# programming: Algotester Lab 6

* Планований час на реалізацію 4 год

Програма №6 Lab# programming: Class Practice 1

* Планований час на реалізацію 2 год

Програма №7 Lab# programming: Class Practice 2

* Планований час на реалізацію 2 год

## **3. Конфігурація середовища до виконання завдань:**



Task.json

## **4. Код програм з посиланням на зовнішні ресурси:**

#include <iostream>

#include <cstring>

#include <cstdio>

#include <algorithm>

using namespace std;

int main() {

char line[256];

std::string sw;

cout << "Enter a line of text: ";

fgets(line, sizeof(line), stdin);

char\* word = strtok(line, " ");

while(word != NULL)

{

string sw = word;

std::sort(sw.begin(), sw.end());

cout << sw << " ";

word = strtok(NULL, " ");

}

return 0;

}

VNSLab6

#include <iostream>

#include <fstream>

#include <cstring>

using namespace std;

struct Record{

char name[40];

char Author[40];

int duration;

int price;

};

void formFile(Record& record);

void printFile();

void deleteEl(int lenth);

void addEl(Record& record, int number);

int main(){

Record Paranoid;

Paranoid = {"Paranoid", "Black Sabbath", 42, 1000};

formFile(Paranoid);

Record Moon;

Moon = {"Dark side of the moon", "Pink floyd", 43, 1500};

formFile(Moon);

Record Rust;

Rust = {"Rust in peace", "Megadeth", 41, 1200};

formFile(Rust);

printFile();

cout << "Enter the lenth of record, which you want to delete" << endl;

int dellenth;

//cin.ignore();

cin >> dellenth;

deleteEl(dellenth);

cout << "Records after deleting" << endl;

printFile();

cout << " " << endl;

Record Wall;

Wall = {"The Wall", "Pink Floyd", 81, 1300};

Record Doors;

Doors = {"The doors", "The Doors", 44, 1400};

int pos;

cout << "enter the after position" << endl;

cin >> pos;

addEl(Wall, pos);

addEl(Doors, pos);

printFile();

fstream file("Records.dat", fstream::out | fstream::trunc | fstream::binary);

file.close();

return 0;

}

void formFile(Record& record)

{

fstream outfile("Records.dat", fstream::out | fstream::app | fstream::binary);

if(outfile.is\_open())

{

outfile.write((char\*)&record, sizeof(Record));

outfile.close();

}

else

{

cout << "error opening file" << endl;

return;

}

}

void printFile()

{

fstream infile("Records.dat", fstream::in | fstream::binary);

if(infile.is\_open())

{

Record record;

while(infile.read((char\*)&record, sizeof(Record)))

{

cout << "Name: " << record.name << endl;

cout << "Author: " << record.Author << endl;

cout << "Lenth: " << record.duration << endl;

cout << "Price: " << record.price << endl;

cout <<"===========" << endl;

}

infile.close();

}

else

{

cout << "error opening file" << endl;

return;

}

}

void deleteEl(int lenth)

{

fstream infile("Records.dat", fstream::binary | fstream::in);

fstream outfile("out.dat", fstream::binary | fstream::out);

if(infile.is\_open() && outfile.is\_open())

{

Record record;

bool founded = false;

while (infile.read((char\*)&record, sizeof(Record))){

if(record.duration!=lenth){

outfile.write((char\*)&record, sizeof(Record));

} else{

founded = true;

}

}

infile.close();

outfile.close();

if(founded)

{

fstream infile("Records.dat", fstream::binary | fstream::out);

fstream outfile("out.dat", fstream::binary | fstream::in);

if(outfile.is\_open() && infile.is\_open())

{

while(outfile.read((char\*)&record, sizeof(Record)))

{

infile.write((char\*)&record, sizeof(Record));

}

outfile.close();

infile.close();

}

else

{

cout << "error2 opening file" << endl;

return;

}

}

else

{

cout << "there is no records with this lenth" << endl;

return;

}

}

else

{

cout << "error opening files" << endl;

return;

}

return;

}

void addEl(Record& record, int number)

{

fstream infile("Records.dat", fstream::binary | fstream::in);

fstream outfile("out.dat", fstream::binary | fstream::out);

if(infile.is\_open() && outfile.is\_open())

{

int current\_number = 0;

Record rec;

while(infile.read((char\*)&rec, sizeof(Record)))

{

if(current\_number==number)

{

outfile.write((char\*)&record, sizeof(Record));

}

outfile.write((char\*)&rec, sizeof(Record));

current\_number++;

}

if(current\_number==number)

{

outfile.write((char\*)&record, sizeof(Record));

}

infile.close();

outfile.close();

fstream infile("Records.dat", fstream::binary | fstream::out);

fstream outfile("out.dat", fstream::binary | fstream::in);

if(outfile.is\_open() && infile.is\_open()){

while(outfile.read((char\*)&record, sizeof(Record)))

{

infile.write((char\*)&record, sizeof(Record));

}

outfile.close();

infile.close();

}

else

{

cout << "error opening file" << endl;

return;

}

}else{

cout << "error opening files" << endl;

return;

}

}

VNSLab8

#include <iostream>

#include <string>

#include <fstream>

using namespace std;

int n;

void create()

{

fstream f("f1.txt", fstream::out);

string text;

cout << "Enter F1 text" << endl;

for(int i = 0; i < 10; i++)

{

getline(cin, text);

f << text << endl;

}

f.close();

}

int golos()

{

fstream f("f1.txt", fstream::in);

if(f.is\_open())

{

string text;

int e, index = 0;

int max\_count = 0, max\_index = 0;

while (!f.eof())

{

e = 0;

text = "";

getline(f, text);

for(int i = 0; i < text.size(); i++)

{

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

if(text[i] == 'a' || text[i] == 'e' || text[i] == 'u' || text[i] == 'i' || text[i] == 'o')

{

e++;

}

}

if(e >= max\_count)

{

max\_count = e;

max\_index = index;

}

index++;

}

cout << max\_index << endl;

return max\_index;

f.close();

}

else

{

cout << "file does not exist" << endl;

}

};

void copyLine(int n)

{

fstream f1("f1.txt", fstream::in);

fstream f2("f2.txt", fstream::out | fstream::app);

if(f1.is\_open() && f2.is\_open())

{

string line;

int index=0;

while(getline(f1, line))

{

if(index != n)

{

f2 << line << endl;

}

++index;

}

f1.close();

f2.close();

} else{

cout << "error" << endl;

return;

}

};

void printFile(const char\* name){

fstream f(name, fstream::in);

if(f.is\_open()){

string line;

while (!f.eof()){

line = "";

getline(f, line);

cout << line << endl;

}

f.close();

} else{

cout << "file does not exist1" << endl;

return;

}

};

int main()

{

create();

cout << endl << "created file:" << endl;

printFile("f1.txt");

n=golos();

copyLine(n);

}

VNSLab9

#include <iostream>

#include <algorithm>

#include <vector>

using namespace std;

bool compareDecreasing(int a, int b) {

return a > b;

}

int main()

{

int N;

cin >> N;

vector<int> a (N);

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

{

cin >> a[i];

}

int k0=0;

int k1=0;

int k2=0;

std::stable\_partition(a.begin(), a.end(), [](int x)

{

return x%3==2;

});

std::stable\_partition(a.begin(), a.end(), [](int x)

{

return x%3==1;

});

std::stable\_partition(a.begin(), a.end(), [](int x)

{

return x%3==0;

});

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

if (a[i]%3 == 0)

{

k0+=1;

k1+=1;

k2+=1;

}

else if (a[i]%3 == 1)

{

k1+=1;

k2+=1;

}

else if (a[i]%3 == 2)

{

k2+=1;

}

}

sort(a.begin(),a.begin()+k0);

sort(a.begin()+k0,a.begin()+k1,compareDecreasing);

sort(a.begin()+k1,a.begin()+k2);

auto new\_a = unique(a.begin(),a.end());

a.erase(new\_a, a.end());

cout<< a.size()<<endl;

for (const auto& an : a) {

std::cout << an << " ";

}

}

Algotester Lab4 Варіант 1

#include <iostream>

using namespace std;

void merge(int arr[], int left, int mid, int right) {

int n1 = mid - left + 1;

int n2 = right - mid;

int leftHalf[n1];

int rightHalf[n2];

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

leftHalf[i] = arr[left + i];

}

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

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

}

int i = 0;

int j = 0;

int k = left;

while (i < n1 && j < n2) {

if (leftHalf[i] <= rightHalf[j]) {

arr[k] = leftHalf[i];

i++;

} else {

arr[k] = rightHalf[j];

j++;

}

k++;

}

while (i < n1) {

arr[k] = leftHalf[i];

i++;

k++;

}

while (j < n2) {

arr[k] = rightHalf[j];

j++;

k++;

}

}

void mergeSort(int arr[], int left, int right) {

if (left < right) {

int mid = left + (right - left) / 2;

mergeSort(arr, left, mid);

mergeSort(arr, mid + 1, right);

merge(arr, left, mid, right);

}

}

void reverseArray(int arr[], int size) {

int start = 0;

int end = size - 1;

while (start < end) {

std::swap(arr[start], arr[end]);

start++;

end--;

}

}

int main()

{

int N;

cin >> N;

int \*a = new int [N];

int \*a0 = new int [N];

int \*a1 = new int [N];

int \*a2 = new int [N];

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

cin >> a[i];

}

int j=0;

int k1=0;

int k2=0;

int k3=0;

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

if (a[i]%3==0){

a0[j]=a[i];

j++;

k1++;

}

}

j=0;

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

if (a[i]%3==1){

a1[j]=a[i];

j++;

k2++;

}

}

j=0;

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

if (a[i]%3==2){

a2[j]=a[i];

j++;

k3++;

}

}

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

}

mergeSort(a0, 0, k1-1);

mergeSort(a1, 0, k2-1);

reverseArray(a1,k2);

mergeSort(a2, 0, k3-1);

j=0;

for(int i = 0; i < k1; i++){

a[j]=a0[i];

j++;

}

for(int i = 0; i < k2; i++){

a[j]=a1[i];

j++;

}

for(int i = 0; i < k3; i++){

a[j]=a2[i];

j++;

}

int new\_N = N;

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

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

if(a[i]==a[j]){

new\_N -= 1;

int \*arr = new int [new\_N];

for(int k = 0, ind=0; k < new\_N; k++, ind++){

if(ind == i){

k--;

} else{

arr[k]=a[ind];

}

}

delete[] a;

a = arr;

N = new\_N;

i--;

j--;

}

}

}

cout << N << endl;

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

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

}

cout << endl;

}

Algotester Lab4 варант 2

#include <iostream>

#include <string>

#include <algorithm>

#include <cstdlib>

using namespace std;

int N;

int main()

{

cin >> N;

int grid[8][8];

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

string temp;

cin >> temp;

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

grid[i][j]=temp[j] - '0';

}

}

int Q;

cin >> Q;

int k[Q][2];

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

{

cin >> k[i][0];

cin >> k[i][1];

k[i][0]--, k[i][1]--;

}

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

{

int nums[N];

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

nums[i]=(i+1);

}

int c=N;

if (grid[k[i][0]][k[i][1]] != 0)

{

cout << '1' << endl;

cout << grid[k[i][0]][k[i][1]] << endl<<endl;

continue;

}

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

if (grid[k[i][0]][j] != 0) {

nums[grid[k[i][0]][j]-1]=0;

c--;

}

if (grid[j][k[i][1]] != 0){

nums[grid[j][k[i][1]]-1]=0;

c--;

}

}

cout <<c << endl;

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

if (nums[j]!=0){

cout << j+1 << " ";

}

}

cout << endl;

cout << endl;

}

}

Algotrester Lab6

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

enum FileOpResult{Success, Failure};

FileOpResult write\_to\_file(char \*name, char \*content);

int main()

{

char name[] = "file1.txt";

char content[100];

cout << "Enter what you want to write to file" << endl;

cin.getline(content, sizeof(content));

FileOpResult res = write\_to\_file(name, content);

if(res == Success)

{

cout << "Success!" << endl;

int choice;

cout << "Do you want print the file?\n1 - yes\nother - no" << endl;

cin >> choice;

if(choice == 1){

fstream file(name, fstream::in);

if(file.is\_open())

{

string line;

while(getline(file, line))

{

cout << line << endl;

}

}

else

{

cout << "Error" << endl;

}

}

}

else

{

cout << "Operation failure!" << endl;

}

return 0;

}

FileOpResult write\_to\_file(char \*name, char \*content)

{

fstream file(name, fstream::out);

if(file.is\_open())

{

file << content;

file.close();

return Success;

}

else

{

cout << "error" << endl;

return Failure;

}

}

Сlass Practice 1

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

enum FileOpResult{Success, Failure, Notexist};

FileOpResult copy\_file(char \*file\_from, char \*file\_to);

int main(){

char file\_from[20];

char file\_to[20];

cout << "enter the name of file from which you want to copy content:" << endl;

cin.getline(file\_from, sizeof(file\_from));

cout << "enter the name of file in which you want to copy content:" << endl;

cin.getline(file\_to, sizeof(file\_to));

FileOpResult res = copy\_file(file\_from, file\_to);

if(res == Success)

{

cout << "Success" << endl;

int choice;

cout << "Do you want to see a new file?\n1 - yes\nother - no" << endl;

cin >> choice;

if(choice == 1)

{

fstream inFile(file\_to, fstream::in);

if(inFile.is\_open())

{

string line;

while(getline(inFile, line))

{

cout << line << endl;

}

}

else

{

cout << "error" << endl;

}

}

}

else if(res == Notexist)

{

cout << file\_from << " does not exist" << endl;

}

else

{

cout << "operation failure" << endl;

}

return 0;

}

FileOpResult copy\_file(char \*file\_from, char \*file\_to)

{

fstream inFile(file\_from, fstream::in);

if(inFile.fail())

{

return Notexist;

}

if(inFile.is\_open())

{

fstream outFile(file\_to, fstream::out | fstream::app);

if(outFile.is\_open())

{

string line;

while(getline(inFile, line))

{

outFile << line << endl;

}

return Success;

}

else

{

return Failure;

}

}

else

{

return Failure;

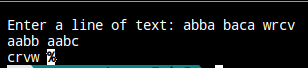
}

}

Class Practice 2

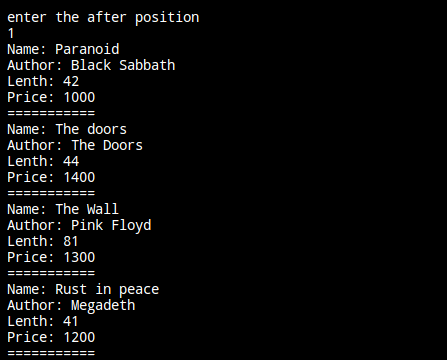
## **5. Результати виконання завдань, тестування та фактично затрачений час:**

VNS Lab6

Час затрачений на виконання завдання 1 год

VNS Lab 6

# 



VNS Lab8

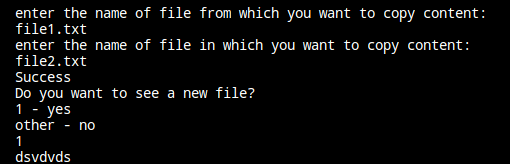
Затрачений час 2 год

# Algotester Lab4

Затрачений час 4 год

# 

# Class practice1



# Algotester Lab 6

# **Висновки:**

При виконанні роботи навчився працювати з файлами, системами числення, бібліотеками.