**ЛАБОРАТОРНАЯ РАБОТА №2**

**«ИССЛЕДОВАНИЕ МЕХАНИЗМА ВИРТУАЛЬНЫХ ФУНКЦИЙ»**

**Цель работы**: Приобретение практических навыков при написании объектно-ориентированных программ с использованием механизмов наследования и виртуальных функций. Освоение особенностей отладки объектно-ориентированных программ.

**Вариант задания**

Для заданной по варианту иерархии описать классы, конструкторы и деструктор, функции ввода и вывода информации на экран. Базовый класс определить как абстрактный, а заданную функцию — как чисто виртуальную в базовом классе и переопределить ее в остальных классах иерархии. Проиллюстрировать корректную работу виртуальных функций и механизма наследования.

Вариант 11

Создать абстрактный базовый класс Поиск с виртуальной функцией поиска. Создать производные классы: символ (char), строка (char \*) и матрица (char \*\*). Функция проверяет, существует ли введенная буква в строке/матрице/равна ли символу?

**2. Код программы на языке C++**

#include <string>

#include <iostream>

using namespace std;

class Base\_search

{

public:

virtual bool find\_elem(const char) = 0;

};

class char\_search : public Base\_search {

const char c;

public:

char\_search(const char c) : c(c) {}

virtual bool find\_elem(const char c) override {

return this->c == c;

}

};

class string\_search : public Base\_search {

const string s;

public:

string\_search(const string s) :s(s){}

virtual bool find\_elem(const char c) override {

for (char e : s)

{

if (e == c) return true;

}

return false;

}

};

class string\_list\_search : public Base\_search {

string\* s;

int s\_count;

public:

string\_list\_search(const string\* s, int s\_count) {

this->s = new string[s\_count];

for (int i = 0; i < s\_count; i++)

this->s[i] = s[i];

this->s\_count = s\_count;

}

virtual bool find\_elem(const char c) override {

for (int i = 0; i < s\_count; i++) {

for (char e : s[i])

{

if (e == c) return true;

}

return false;

}

}

};

int main()

{

system("chcp 1251");

char\_search obj1('L');

string\_search obj2("This is testovay strocka");

cout << "Введите колличестово строк в матрице ";

int s\_count;

cin >> s\_count;

string\* s = new string[s\_count];

for (int i = 0; i < s\_count; i++) {

cout << "Введите строку " << i << " -->";

cin >> s[i];

}

string\_list\_search obj3(s, s\_count); char serch\_el;

while (1) {

cout << "Введите символ для поиска. 0 для выхода ";

cin >> serch\_el;

if (serch\_el == '0') break;

if (obj1.find\_elem(serch\_el)) cout << "Данный символ есть в 1-ом класе"<<endl;

if (obj2.find\_elem(serch\_el)) cout << "Данный символ есть в 2-ом класе" << endl;

if (obj3.find\_elem(serch\_el)) cout << "Данный символ есть в 3-ом класе" << endl;

}

}

**3. Тестирование и отладка**

Для первого класса будет передан символ L, для второго класса будет передана строка This is testovay strocka. Для третьего класса будет сделан вво данных с клавиатуры.

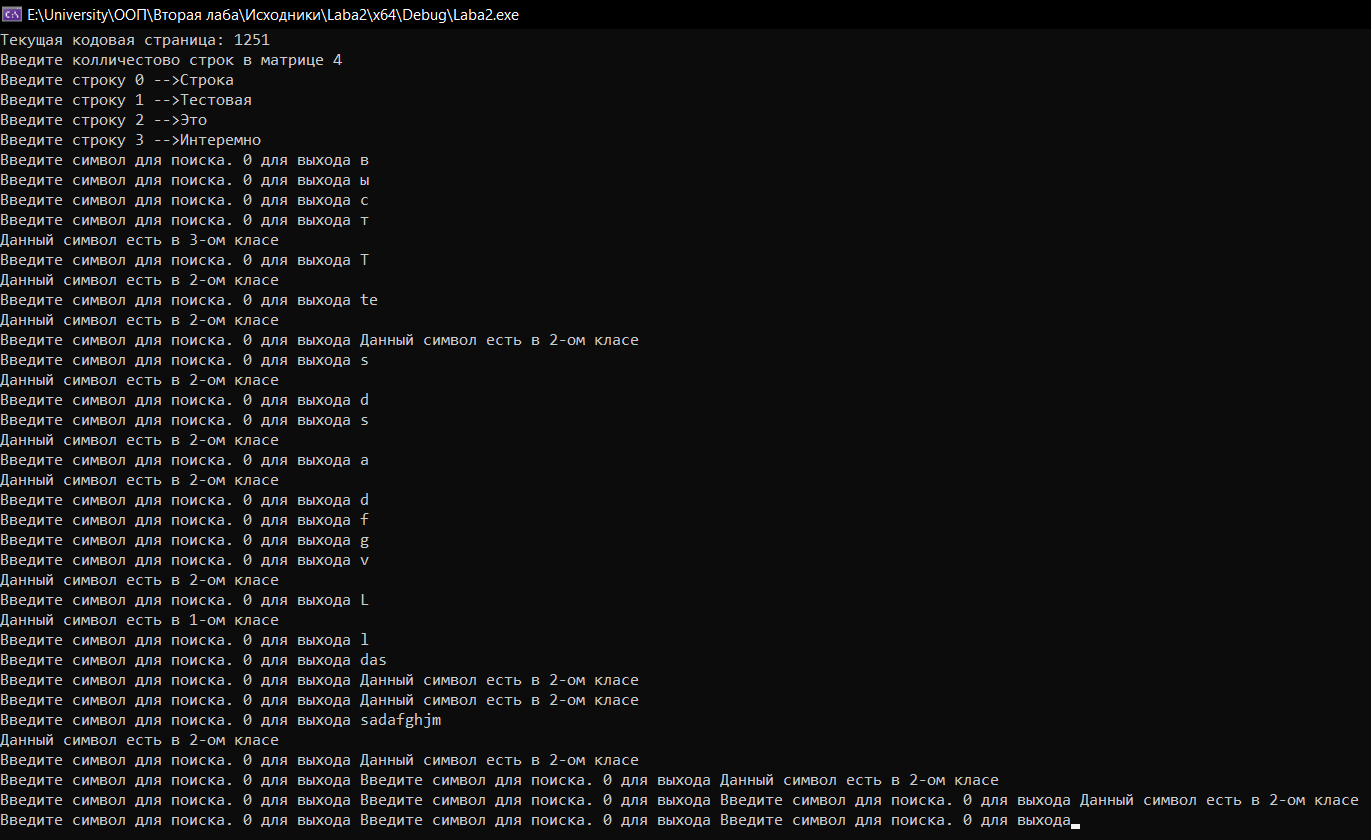


Рисунок 1 – Ввода стартовых данных и тестирования

После ввода данных, программа запрашивает ввод символа, после его ввода, она проверяет его, и по итогу , потом просто вызывает методы у класса.

В результате тестирования, видно, что классы работают правильно, виртуальный метод и поля наследуются корректно.

**Вывод**

При выполнении данной лабораторной работы были приобретены практических навыков при написании объектно-ориентированных программ с использованием механизмов наследования и виртуальных функций. Освоены особенностей отладки объектно-ориентированных программ.