# 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.

Задания

1. Изучить основы работы с механизмами наследования и виртуальных функций;
2. Разработать программу на языке C++ согласно варианту 11 методических указаний (Рисунок 1). Для заданной 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 её в остальных классах иерархии;

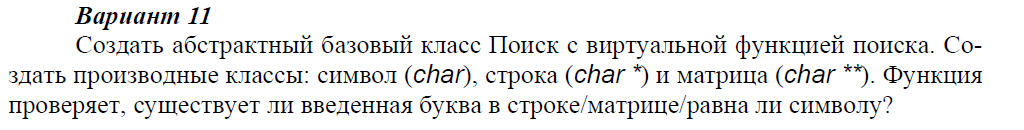


Рисунок 1 – Вариант задания

1. Разработать тестовые примеры и выполнить отладку программы. Проиллюстрировать корректную работу виртуальных функций и механизма наследования;
2. Получить результаты работы программы и исследовать её свойства для различных режимов работы, сформулировать выводы;

Текст программы

#include <iostream>

#include <cstring>

#include <conio.h>

using namespace std;

// Абстрактный класс для char, char\* и char\*\*

class Srch {

public:

Srch() {} // конструктор

virtual ~Srch() { // деструктор

cout << "Destructor worked \n";

};

virtual void put() = 0; // ввод объекта

virtual void print() = 0; // вывод объекта

virtual bool search(char c) = 0; // поиск символа в объекте

};

// Класс для символов

class Char : public Srch {

char c;

public:

Char() { // конструктор

c = '\0';

}

~Char() {} // деструктор

void put() { // ввод символа

cout << "Enter symbol - ";

cin >> c;

}

void print() { // вывод символа

cout << c << endl;

}

bool search(char c1) { // сравнение с другим символом

if (c1 == c) return true;

else return false;

}

};

// Класс для строк

class String : public Srch {

char\* str;

public:

String() { // конструктор

str = new char[100];

}

~String() { // деструктор

delete[] str;

str = NULL;

}

void put() { // ввод строки

cout << "Enter string - ";

cin >> str;

}

void print() { // вывод строки

cout << str << endl;

}

bool search(char c) { // поиск символа в строке

if (strchr(str, c))

return true;

else return false;

};

};

// Класс для матриц из строк

class Matrix : public Srch {

char\*\* matr; // указатель на матрицу

int count; // число строк матрицы

public:

Matrix() { // конструктор

matr = new char\* [100];

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

matr[i] = new char[100];

count = 0;

}

~Matrix() { // деструктор

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

delete[] matr[i];

delete[] matr;

count = 0;

}

void put() { // ввод матрицы

count = 0;

cout << "Enter strings of matrix:" << endl;

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

{

cout << i + 1 << ") Enter string - ";

cin >> matr[i];

count++;

// продолжить / прекратить ввод строк

cout << "continue? 1 - YES / 0 - NO" << endl << endl;

char ask = ' ';

while ((ask != '1') && (ask != '0'))

ask = \_getch();

if (ask == '0') break;

}

}

void print() { // вывод матрицы

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

cout << matr[i] << endl;

}

bool search(char c = ' ') { // поиск символа в матрице

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

if (strchr(matr[i], c))

return true;

}

return false;

}

};

int main() {

char symb;

Srch\* obj1 = new Char();

Srch\* obj2 = new String();

Srch\* obj3 = new Matrix();

// Проверка работы класса Char

obj1->put();

cout << "Enter symbol searched in Char object - ";

cin >> symb;

cout << "Result - " << obj1->search(symb) << "\n\n";

// Проверка работы класс String

obj2->put();

cout << "Enter symbol searched in String object - ";

cin >> symb;

cout << "Result - " << obj2->search(symb) << "\n\n";

// Проверка работы класса Matrix

obj3->put();

cout << "Enter symbol searched in Matrix object - ";

cin >> symb;

cout << "Result - " << obj3->search(symb) << "\n\n";

// Проверка вывода данных

obj1->print();

obj2->print();

obj3->print();

// Очистка памяти под указатели

cout << endl << "PROGRAMM FINISHED" << endl;

delete obj1;

delete obj2;

delete obj3;

return 0;

}

Тестовые примеры

Написанная программа по очереди предлагает пользователю ввести объекты трёх классов, ввести символ и затем выполнить поиск этого символа в ведённый объект.

Для начала был выполнен ввод объектов классов и произведена попытка поиска в них символов, которые в них отсутствует. В результате методами search классов (осуществляющих поиск символа в объекте) было возвращено значение false, т.е. 0 (Рисунок 2).

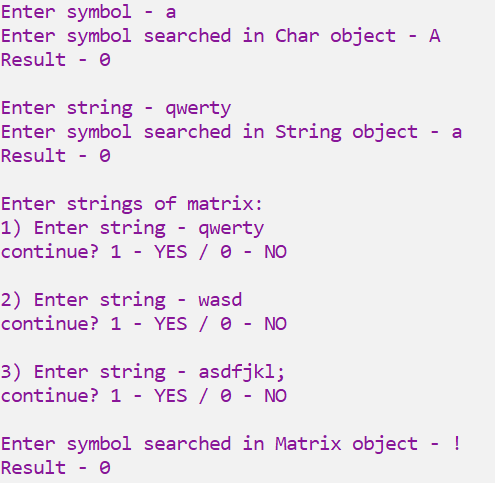


Рисунок 2 – Проверка ввода данных и поиска символа

Тот же метод был проверен для символов, присутствующих в объектах классов. В результате методом было возвращено значение true, т.е. 1 (Рисунок 3).

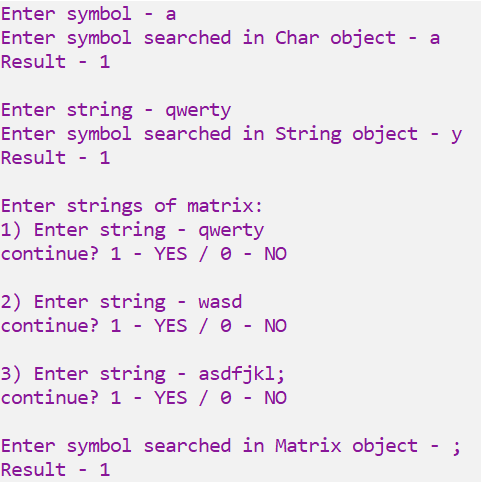


Рисунок 3 – Тестирование поиска символа в объекте класса

Затем программой были запущены методы вывода ключевых полей классов. В результате на экран были выведены все ранее введённые символы и строки (Рисунок 4).

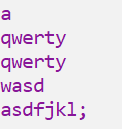


Рисунок 4 – Вывод полей классов

Перед окончание программы была очищена память под указатели на объекты классов, в результате чего были запущены деструкторы классов (Рисунок 5).

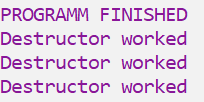


Рисунок 5 – Вызов деструкторов классов при очищении памяти

Вывод

В ходе работы были исследованы механизмы реализации наследования и полиморфизма в объектно-ориентированном программировании. Была написана программа, осуществляющая работу с объектами классов, производных от абстрактного с применением виртуальных функций.