Министерство образования и науки РФ

Севастопольский государственный университет

Кафедра информатики и управления в технических системах

ОТЧЕТ

по лабораторной работе №2

ИССЛЕДОВАНИЕ МЕХАНИЗМА ПРОСТОГО НАСЛЕДОВАНИЯ

по дисциплине «Объектно-ориентированное программирование»

Выполнил:

Студент группы ИВТ/б 22-о

Черняев Н.Г.

Проверил:

Тимофеев И.С.

г. Севастополь 2018

**Цель работы**

Исследование основных средств создания базового и порожденного классов. Исследование особенностей вызова методов порожденного класса при простом наследовании.

**Вариант №10**

Базовый класс Символ с полем «символ» и методом «Печать на экран». Производный класс Строка с полем Слово (char \*). Переопределить для него функцию Печать так, чтобы она выводила слово на экран. Описать в классе дополнительную функцию Вхождение, определяющую, есть ли Символ в Слове.

Описать заданный по варианту базовый класс минимум с одним информационным полем. Для базового класса определить два конструктора, а также два метода: метод ввода данных и метод, указанный в варианте задания.

Описать класс-наследник, расширив его функциональность относительно базового класса в соответствии с вариантом задания – добавив поля и методы, свойственные этому классу. Определить в классе конструкторы, функции ввода данных и вывода данных на экран. Переопределить один из методов базового класса в классе-наследнике.

Создать объекты базового и производного классов, ввести необходимые данные. Для порожденного объекта осуществить вызов переопределенного метода и вызов этого же метода из базового класса.

**Текст программы**

#include <string>

#include <iostream>

using namespace std;

class Symbol {

protected:

char symb;

public:

Symbol(char \_symb) { symb = \_symb; };

Symbol() { cout << "Enter symbol" << endl; cin >> symb; };

char GetSymb() { return symb; };

void Output() { cout << "Symbol= " << GetSymb() << endl; };

};

class Line: public Symbol {

protected:

string word;

public:

Line(char \_symb) : Symbol(\_symb) {};

Line() : Symbol() { cout << "Enter word= " << endl; cin >> word; };

Line(string \_word) { word = \_word; };

void Output() { cout << "Word=" << word << endl; };

void Include(Symbol x);

};

void Line::Include(Symbol x) {

int i, k = 0;

char y = x.GetSymb();

while (word[i] != '\0') {

if (word[i] == y) {

k = 1;

}

i++;

};

if (k == 1) {

cout << "Word includes symb" << endl;

}

else {

cout << "Word doesnt include symb" << endl;

};

};

int main()

{

Symbol Symbol1('q');

Line Line1;

Symbol1.Output();

Line1.Output();

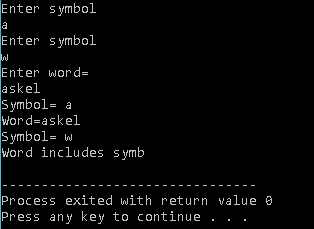
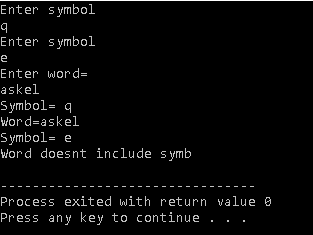
Line1.Symbol::Output();

Line1.Include(Symbol1);

return 0;

}

**Тестовые примеры:**

Вывод:

В данной лабораторной работе были исследованы основные средства создания базового и порожденного классов и особенности вызова порожденного класса при простом наследовании.