Федеральное государственное бюджетное образовательное учреждение высшего

образования

«Чувашский государственный университет им. И.Н. Ульянова»

Кафедра вычислительной техники

Лабораторная работа № 7

Наследование. Полиморфизм.

Вариант 9

Выполнил: Иванов В.С.

студент группы ИВТ-41-22

Проверил: кандидат технических наук

Обломов Игорь Александрович

Чебоксары, 2024

Цель работы: изучение наследования и полиморфизма, применение их на практике

Выполнение работы:

***Задание №1-4.***

Объявить базовый и производный классы, моделирующие слово (базовый класс) и строку (производный класс). В базовом классе предусмотреть поле для хранения одного слова произвольной длины (можно использовать тип string), а в производном, кроме наследованного слова, необходимо объявить поле, содержащее количество слов в строке. Для классов определить методы, обеспечивающие ввод слов и строк и вывод их на экран. В конструкторах базового и производного классов обеспечить вывод в стандартный поток сообщения, идентифицирующего принадлежность объекта тому или иному классу. Дополните объявление класса слово статическим полем, приведите пример его использования объектами и методами производного класса строка. В классе слово определить дружественную функцию, выводящую отдельное слово в стандартный поток.

class Word {

string word;

int first, second;

public:

Word(int f, int s) :first(f), second(s) {};

Word(string w) :word(w) {};

int Get() {

cout << "Object address:" << this << " ";

return first;

cout << endl;

}

friend void showWord(Word& w);

void inputWord() {

cout << "Enter word: ";

cin >> word;

}

static int co;

static int Get\_Count() {

return co;

}

~Word() = default;

};

void showWord(Word& w) {

cout << "Friend void " << w.word << endl;

}

int Word::co = 20;

class WordCount : public Word {

int wordCount;

//long tt;

public:

WordCount(int f, int s) :Word(f, s) {};

WordCount(string w) :Word(w) {

wordCount = 1;

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

if (w[i] == ' ') {

wordCount++;

}

}

}

void showWordCount() {

showWord(\*this);

cout << wordCount << endl;

}

static void Set\_Count() {

co = 0;

}

~WordCount()=default;

};

Int main(){

Word w\_1("Pupupu");

showWord(w\_1);

WordCount w\_2("Pupupupu pupupu");

w\_2.showWordCount();

}

***Задание №5.***

Создать иерархию классов, показать пример наследования переменной this объектами производных классов.

class Word {

string word;

int first, second;

public:

Word(int f, int s) :first(f), second(s) {};

};

void showWord(Word& w) {

cout << "Friend void " << w.word << endl;

}

int Word::co = 20;

class WordCount : public Word {

int wordCount;

//long tt;

public:

WordCount(int f, int s) :Word(f, s) {};

};

class Test\_3 :public Word {

//char tst;

public:

Test\_3(int f, int s) :Word(f, s) {};

};

class All :public WordCount, public Test\_3 {

public:

All(int f, int s, int t, int u):WordCount(f, s), Test\_3(t, u) {};

};

All a(1, 2, 3, 4);

***Задание №6-9.***

Разработать иерархию классов «человек, служащий, студент», в которой класс человек имеет поля имя, фамилия, возраст. Класс служащий дополняет его полем специальность, а класс студент – полями группа и средний балл. Предусмотреть полиморфные методы, позволяющие получить информацию о субъекте в зависимости от его типа. В задании №6 класс человек определить как абстрактный. Привести примеры использования абстрактного класса. В иерархии «Человек, служащий, студент» преобразовать указатель на класс человек в указатель на класс служащий и студент. В соответствие с заданием №7 обеспечить понижающее преобразование ссылок класса человек в ссылку на класс служащий, ссылки на класс служащий в ссылку на класс студент.

class Pupupu {

public:

virtual void show() = 0;

};

class Human: public Pupupu {

string name;

string surname;

int age;

public:

Human() {};

Human(string n, string s, int a) :name(n), surname(s), age(a) {};

void show() {

cout << "Human:" << name << " " << surname << " " << age << endl;

}

};

class Employee :public Human {

string speciality;

public:

Employee() {};

Employee(string n, string s, int a, string sp) :Human(n, s, a) {

speciality = sp;

}

void show() {

cout << "Employee:" << speciality << endl;

}

};

class Student :public Employee {

string group;

float avg\_mark;

public:

Student() {};

Student(string n, string s, int a, string g, float m) :Employee(n, s, a, g) {

group = g;

avg\_mark = m;

}

void show() {

cout << "Student:" << group << " " << avg\_mark << endl;

}

};

Int main(){

Human\* h\_1 = new Human;

Employee\* emp\_1 = new Employee();

Student\* st\_1 = new Student();

Employee emp("Vladimir","Ivanov", 19, "IT");

h\_1=&emp;

h\_1->show();

Student st("Vladimir","Ivanov", 19, "IT-21", 5.5);

h\_1=&st;

h\_1->show();

h\_1 = dynamic\_cast<Employee\*>(h\_1);

h\_1->show();

h\_1 = dynamic\_cast<Student\*>(h\_1);

h\_1->show();

Employee emp;

Employee& emp\_ref = emp;

Human& h\_ref = emp;

}

Вывод: изучил наследование и полиморфизм, применил их на практике