Министерство образования и науки Российской Федерации

Севастопольский государственный университет

Институт информационных технологий и управления в технических системах

ОТЧЕТ

по лабораторной работе №3

МНОЖЕСТВЕННОЕ НАСЛЕДОВАНИЕ.

ДРУЖЕСТВЕННЫЕ КЛАССЫ И ФУНКЦИИ.

Выполнил: ст. гр. ИТб-21

Куркчи А.Э.

Проверила:

Тимофеев И.С.

Севастополь

2015

# 1. Цель

Приобретение практических навыков при написании объектно-ориентированных программ. Освоение особенностей отладки объектно-ориентированных программ. Изучение механизма множественного наследования. Приобретение навыков работы с механизмом «дружественности».

2. Постановка задач

Описать интерфейс ввода-вывода. Описать иерархию классов, заданную по варианту. Для каждого класса описать конструкторы и деструктор (по необходимости), функции ввода и вывода значений полей. В каждом классе должны присутствовать минимум одно уникальное поле и один уникальный метод. Проиллюстрировать корректную работу механизма множественного наследования. Класс-наследник должен реализовать описанный ранее интерфейс. Описать дружественную функцию для класса-наследника и заданного по варианту класса.

Вариант №1

Базовые классы: Юноши (ФИО, год рождения), Учащийся (№зачетки, курс, ср.бал). Класс-наследник: Студент.

Дружественность: класс Ведомость.

3. Текст программы

#include <iostream>

#include <stdio.h>

#include <string.h>

#include <math.h>

**using** **namespace** std**;**

const int STR\_SIZE **=** 128**;**

class student**;**

class sheet**;**

bool is\_good**(**student**\*);**

class man **{**

private**:**

char name**[**STR\_SIZE**];**

int birth\_year**;**

int age**;**

public**:**

man**()** **{**

strcpy**(**name**,**"empty"**);**

birth\_year **=** 1970**;**

age **=** 1**;**

**}**

man**(**char n**[],**int by**,**int a**)** **{**

strcpy**(**name**,**n**);**

birth\_year **=** by**;**

age **=** a**;**

**}**

virtual void scan**()** **{**

cout **<<** "Input name:" **<<** endl**;**

cin**.**getline**(**name**,**STR\_SIZE**);**

cout **<<** "Input birth year:" **<<** endl**;**

cin **>>** birth\_year**;**

cout **<<** "Input age:" **<<** endl**;**

cin **>>** age**;**

**}**

virtual void print**()** **{**

cout **<<** "Name: " **<<** name **<<** endl**;**

cout **<<** "Birth year: " **<<** birth\_year **<<** endl**;**

cout **<<** "Age: " **<<** age **<<** endl**;**

**}**

void grow**()** **{**

age**++;**

**}**

friend class sheet**;**

friend bool is\_good**(**student**\*);**

**};**

class learner **{**

private**:**

int id**;**

int course**;**

double avarage**;**

public**:**

learner**()** **{**

id **=** 1**;**

course **=** 1**;**

avarage **=** 5.0**;**

**}**

learner**(**int i**,**int c**,**double a**)** **{**

id **=** i**;**

course **=** c**;**

avarage **=** a**;**

**}**

virtual void scan**()** **{**

cout **<<** "Input id:" **<<** endl**;**

cin **>>** id**;**

cout **<<** "Input course:" **<<** endl**;**

cin **>>** course**;**

cout **<<** "Input avarage:" **<<** endl**;**

cin **>>** avarage**;**

**}**

virtual void print**()** **{**

cout **<<** "ID: " **<<** id **<<** endl**;**

cout **<<** "Course: " **<<** course **<<** endl**;**

cout **<<** "Avarage: " **<<** avarage **<<** endl**;**

**}**

void pass**()** **{**

course**++;**

**}**

friend class sheet**;**

friend bool is\_good**(**student**\*);**

**};**

class student**:** public man**,** public learner **{**

private**:**

int dormitory**;**

public**:**

student**():**man**(),**learner**()** **{**

dormitory **=** 1**;**

**}**

student**(**char n**[],**int by**,**int a**,**int i**,**int c**,**double av**,**int d**):**man**(**n**,**by**,**a**),**learner**(**i**,**c**,**av**)** **{**

dormitory **=** d**;**

**}**

virtual void scan**()** **{**

man**::**scan**();**

learner**::**scan**();**

cout **<<** "Input dormitory:" **<<** endl**;**

cin **>>** dormitory**;**

**}**

virtual void print**()** **{**

man**::**print**();**

learner**::**print**();**

cout **<<** "Dormitory: " **<<** dormitory **<<** endl**;**

**}**

void resettle**(**int d**)** **{**

dormitory **=** d**;**

**}**

friend class sheet**;**

friend bool is\_good**(**student**\*);**

**};**

class sheet **{**

private**:**

int id**;**

student**\*** stud**;**

public**:**

sheet**(**student**\*** s**)** **{**

stud **=** s**;**

**}**

sheet**(**student**\*** s**,**int i**)** **{**

stud **=** s**;**

id **=** i**;**

**}**

void scan**()** **{**

cout **<<** "Input sheet id:" **<<** endl**;**

cin **>>** id**;**

**}**

void print**()** **{**

cout **<<** "\tSheet #" **<<** id **<<** endl**;**

cout **<<** "#" **<<** stud**->**id **<<** " " **<<** stud**->**name **<<** **(**is\_good**(**stud**)** **?** " good" **:** " not good"**)** **<<** " avarage " **<<** stud**->**avarage **<<** endl**;**

**}**

**};**

bool is\_good**(**student**\*** stud**)** **{**

**return** **(**stud**->**avarage **>** 4.0**);**

**}**

int main**()** **{**

cout **<<** "SevGU semestr.03 OOP lab.03" **<<** endl**;**

cout **<<** "Demonstration start" **<<** endl**;**

student**\*** stud**;**

sheet**\*** sh**;**

// cout << "Creating student and assigning it to stud" << endl;

cout **<<** "> stud = new student();" **<<** endl**;**

stud **=** **new** student**();**

// cout << "Scanning student" << endl;

cout **<<** "> stud->scan();" **<<** endl**;**

stud**->**scan**();**

// cout << "Growing student" << endl;

cout **<<** "> stud->grow();" **<<** endl**;**

stud**->**grow**();**

// cout << "Passing student" << endl;

cout **<<** "> stud->pass();" **<<** endl**;**

stud**->**pass**();**

// cout << "Resettling student in 4th dorm" << endl;

cout **<<** "> stud->resettle(4);" **<<** endl**;**

stud**->**resettle**(**4**);**

// cout << "Printing student" << endl;

cout **<<** "> stud->print();" **<<** endl**;**

stud**->**print**();**

// cout << "Creating sheet for student and assigning it to sh" << endl;

cout **<<** "> sh = new sheet(stud);" **<<** endl**;**

sh **=** **new** sheet**(**stud**);**

// cout << "Scanning sheet" << endl;

cout **<<** "> sh->scan();" **<<** endl**;**

sh**->**scan**();**

// cout << "Printing sheet" << endl;

cout **<<** "> sh->print();" **<<** endl**;**

sh**->**print**();**

cout **<<** "Demonstration end" **<<** endl**;**

**return** 0**;**

**}**

4. Тестовые примеры

На рисунке 1 представлен вывод программы

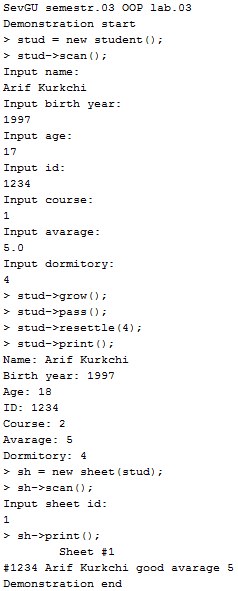


Рисунок 1 – Вывод программы

После ввода 3х событий начинается демонстрация корректности программы. В строках, начинающихся с «>», печатается выполняемая в программном коде строка, в остальных строках – вывод программы. Метод scan отвечает за чтение параметров из консоли, а print их печать. Метод grow отвечает за увеличение возраста, pass за переход на следующий курс, а resettle за переселение студента.

Вывод

В ходе лабораторной работы были приобретены практические навыки написания объектно-ориентированных программ. Освоены особенности отладки объектно-ориентированных программ. Приобретены навыки работы с механизмом «дружественности».