Министерство образования Республики Беларусь

Учреждение образования

«Брестский Государственный технический университет»

Кафедра ИИТ

**Лабораторная работа №5**

По дисциплине «Проектирование программ в ИС»

Тема: **Иерархии классов. Наследование”**

**Выполнил:**

Студент 2 курса

Группы ИИ-21

Парфеевец И.А.

**Проверил:**

Монтик Н.С.

Брест 202

ЦЕЛЬ РАБОТЫ

1. Изучение правил наследования классов.

2. Реализация одиночного наследования классов.

3. Изучение управления методами и свойствами производных классов через объекты производных классов и через указатели на объекты производных классов.

4. Изучение правил описания наследования и диаграмм классов в языке UML.

**Код программы:**

#include <iostream>

using std::cout;

using std::string;

using std::endl;

//создадим класс работников (основополагающий класс)

class DevicesInOffice {

public:

string monitor;

string CPU;

string GPU;

DevicesInOffice() {

monitor = "wasn't specified";

CPU = "wasn't specified";

GPU = "wasn't specified";

}

DevicesInOffice(string CPU, string monitor, string GPU) {

this->CPU = CPU;

this->GPU = GPU;

this->monitor = monitor;

}

DevicesInOffice(DevicesInOffice& Sample) {

this->CPU = Sample.CPU;

this->GPU = Sample.GPU;

this->monitor = Sample.monitor;

}

string isstrong() {

if (CPU.at(0) == 'P' || CPU.at(0) == 'p') { return "his pc is weak"; }

else return "his pc is good enough";

}

};

class Employee:virtual public DevicesInOffice {

public:

string name;

string company;

int age;

string majority;

Employee(string name, string company, int age, string majority) {

this->age = age;

this->company = company;

this->name = name;

this->majority = majority;

};

Employee(Employee& Sample,DevicesInOffice& Template) {

this->age = Sample.age;

this->company = Sample.company;

this->name = Sample.name;

this->majority = Sample.majority;

this->CPU = Template.CPU;

this->GPU = Template. GPU;

this->monitor = Template.monitor;

}

Employee(Employee& Sample) {

this->age = Sample.age;

this->company = Sample.company;

this->name = Sample.name;

this->majority = Sample.majority;

}

Employee() {

name = "name wasn't entered";

company = "company wasn't entered ";

majority = "majority wasn't entered ";

age = 0;

}

void IntroduceYourself() {

cout << "Hi,my name is " << name << " I'am " <<

age << " Y.O. My majority is " <<majority << ". I work for " << company << " company!\n\n";

}

void Devices() {

cout << name << " is working on PC with those Devices : \n";

cout << "Monitor : " << monitor; cout << endl;

cout << "CPU : " << CPU;; cout << endl;

cout << "GPU : " << GPU; cout << endl << endl;

}

};

class Developer:virtual public Employee, virtual public DevicesInOffice {

public:

string FavoriteProgrammingLanguage;

Developer(string name, string company, int age, string majority,string FavoriteProgrammingLanguage) {

this->age = age;

this->company = company;

this->name = name;

this->majority = majority;

this->FavoriteProgrammingLanguage = FavoriteProgrammingLanguage;

};

Developer(Developer& Sample) {

this->age = Sample.age;

this->company = Sample.company;

this->name = Sample.name;

this->majority = Sample.majority;

this->FavoriteProgrammingLanguage = FavoriteProgrammingLanguage;

}

Developer() {

name = "name wasn't entered";

company = "company wasn't entered ";

majority = "majority wasn't entered ";

FavoriteProgrammingLanguage = " Favorite Programming Language wasn't specified";

age = 0;

}

void InitializeDevices (string CPU, string monitor, string GPU) {

this->CPU = CPU;

this->GPU = GPU;

this->monitor = monitor;

}

void IntroduceYourself() {

cout << "Hi,my name is " << name << " I'am " <<

age << " Y.O. " << majority << ".I work for " << company << " company!\n";

cout << name << "'s favorite Programming language is " << FavoriteProgrammingLanguage<<endl<<endl;

}

};

int main() {

//для начала посмотрим ,как работают методы класса Employee с унаследованными переменными от класса DevicesInOffice

Employee Andre("Andrey", "TaksoPark", 19, "Taksi");

DevicesInOffice AndreysDevices("PENTIUM 4700", "PANASONIC 47 HZ ", "NVIDIA HD GRAPHUCS 630");

Employee Andrey(Andre, AndreysDevices);

Andrey.IntroduceYourself();

Andrey.Devices(); string buff = Andrey.isstrong(); cout << buff << endl<<endl<<endl;

cout << "||||||||||||||||||||||||||||||||||\n\n\n";

//теперь посмотрим , как работают унаследованные методы класса Employee и DevicesInOffice у класса Developer

//и как вызывать метод IntroduceYourself() ,который присутствует в обоих классах

Developer Dandy ("Dandy", "BSTU", 18, "Programmer","Turbo C");

cout << "introduceYourself() from Employee class : " << endl;

Dandy.Employee::IntroduceYourself(); Dandy.Devices();

Dandy.InitializeDevices("IntelCore I5", "Samsung 144 hz ", "Nvidia 1050 Ti");

Dandy.Devices();

buff = Dandy.isstrong(); cout << buff<<endl;

cout<<"||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||\n\n" ;

cout << "introduceYourself() from Developer class :\n " << endl;

Dandy.IntroduceYourself();

return 713;

}

Результат работы программы :

Hi,my name is Andrey I'am 19 Y.O. My majority is Taksi. I work for TaksoPark company!

Andrey is working on PC with those Devices :

Monitor : PANASONIC 47 HZ

CPU : PENTIUM 4700

GPU : NVIDIA HD GRAPHUCS 630

his pc is weak

||||||||||||||||||||||||||||||||||

introduceYourself() from Employee class :

Hi,my name is Dandy I'am 18 Y.O. My majority is Programmer. I work for BSTU company!

Dandy is working on PC with those Devices :

Monitor : Samsung 144 hz

CPU : IntelCore I5

GPU : Nvidia 1050 Ti

his pc is good enough

||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

introduceYourself() from Developer class :

Hi,my name is Dandy I'am 18 Y.O. Programmer.I work for BSTU company!

Dandy's favorite Programming language is Turbo C

**Вывод:** Изучил правила наследования классов