

Министерство образования Республики Беларусь
Учреждение образования
«Брестский государственный технический университет»
Кафедра ИИТ

Лабораторная работа №6.
"Наследование и виртуальные функции"

Выполнил:
Ст. 2 курса гр. АС-53
Альциванович Н.В.
Проверила:
Давидюк Ю. И.

Брест, 2020

1. Цель. Получить практические навыки создания иерархии классов и использования статических компонентов класса.

2. Постановка задачи (Вариант 1)

Написать программу, в которой создается иерархия классов. Включить полиморфные объекты в связанный список, используя статические компоненты класса. Показать использование виртуальных функций.

студент, преподаватель, персона, завкафедрой;

Классы:

- Персона
 - Завкафедрой
 - Преподаватель
 - Студент

Классы выстроены в соответствии с их иерархией.

Конструкторы:

- Без параметров
- Копирования
- С параметрами

Деструктор виртуальный.

Функции:

- Добавления в лист(Вызывается при создании объекта класса)
- Вывода информации
- Возвращения и изменения параметров

3. Иерархия классов в виде графа:

- Person
 - HeadofDed
 - Prepod
 - Student

4. Определение пользовательских классов с комментариями.

//Базовый класс «Персона»

```
#pragma once
#include <iostream>
#include <string>
```

```
using std::cout;
using std::endl;
using std::string;
```

```
class Persona {
protected:
```

```

        int aAge;
        string aName;
public:
    Persona();
    Persona(int, string);
    Persona(const Persona&);
    ~Persona();
    void Push();
    virtual void View() = 0;
};

Persona::Persona(int age, string name) : aAge(age), aName(name) {}
Persona::Persona() : aAge(0), aName("??") {}
Persona::Persona(const Persona& oth) : Persona(oth.aAge, oth.aName) {}
Persona::~~Persona() {}

class PersonList {
    class Node {
    public:
        Node(Persona*);
        Node();
        Node* apNext;
        Persona* apElem;
    };
    static Node* apBegin;
    static Node* apEnd;
public:
    static void PushEnd(Persona*);
    static void ViewList();
};

PersonList::Node* PersonList::apBegin = nullptr;
PersonList::Node* PersonList::apEnd = nullptr;
PersonList::Node::Node() : apNext(nullptr) {}
PersonList::Node::Node(Persona* elem) : apNext(nullptr), apElem(elem) {}
void PersonList::PushEnd(Persona* elem) {
    Node* pTemp = new Node(elem);
    if (!apBegin) apBegin = apEnd = pTemp;
    else {
        apEnd->apNext = pTemp;
        apEnd = pTemp;
    }
}

void PersonList::ViewList() {
    int i = 0;
    for (Node* pCur = apBegin; pCur; pCur = pCur->apNext) {
        cout << ++i << " ";
        pCur->apElem->View();
    }
}

void Persona::Push() {
    PersonList::PushEnd(this);
}

// Класс «Студент»
#pragma once
#include "Persona.h"

class Student :

```

```

        public Persona {
protected:
        int aCourse;
        string aFaculty;
public:
        Student();
        Student(const Student&);
        Student(int, string, string, int);
        void View() override;
};

Student::Student(int age, string name, string facult, int course) :
        Persona(age, name),
        aFaculty(facult),
        aCourse(course)
{}

Student::Student() : Student(0, "???", "???", 0) {}

Student::Student(const Student& oth) :
        Student(oth.aAge, oth.aName, oth.aFaculty, oth.aCourse)
{}

void Student::View() {
        cout << "Student ->\nAge: " << aAge
                << "\nName: " << aName << "\nFaculty: " << aFaculty
                << "\nCourse: " << aCourse << endl << endl;
}

//Класс «Преподаватель»
#pragma once
#include "Persona.h"
class Prepod :
        public Persona
{
protected:
        string aAcademicSubject;
public:
        Prepod();
        Prepod(const Prepod&);
        Prepod(int, string, string);
        ~Prepod();
        void View() override;
};

Prepod::Prepod() : Prepod(0, "???", "???", "") {}
Prepod::Prepod(int age, string name, string academicSubject) :
        Persona(age, name),
        aAcademicSubject(academicSubject)
{}
Prepod::Prepod(const Prepod& oth) :
        Prepod(oth.aAge, oth.aName, oth.aAcademicSubject)
{}

void Prepod::View() {
        cout << "Prepod ->\nAge: " << aAge
                << "\nName: " << aName << "\nAcademic subject: " << aAcademicSubject << endl <<
endl;
}

Prepod::~~Prepod() {}

```

```

//Класс «Завкафедрой»
#pragma once
#include "Persona.h"
class HeadOfDed :
    public Persona
{
protected:
    string aNameOfDep;
public:
    HeadOfDed();
    HeadOfDed(const HeadOfDed&);
    HeadOfDed(int, string, string);
    ~HeadOfDed();
    void View() override;
};

HeadOfDed::HeadOfDed() : HeadOfDed(0, "???", "???", "??") {}
HeadOfDed::HeadOfDed(int age, string name, string nameDep) :
    Persona(age, name),
    aNameOfDep(nameDep)
{}
HeadOfDed::HeadOfDed(const HeadOfDed& oth) :
    HeadOfDed(oth.aAge, oth.aName, oth.aNameOfDep)
{}

void HeadOfDed::View() {
    cout << "head of departament ->\nAge: " << aAge
         << "\nName: " << aName << "\ndepartament: " << aNameOfDep << endl << endl;
}

HeadOfDed::~~HeadOfDed() {}
// Реализация функций классов
PersonList::Node* PersonList::apBegin = nullptr;
PersonList::Node* PersonList::apEnd = nullptr;
PersonList::Node::Node() : apNext(nullptr) {}
PersonList::Node::Node(Persona* elem) : apNext(nullptr), apElem(elem) {}
void PersonList::PushEnd(Persona* elem) {
    Node* pTemp = new Node(elem);
    if (!apBegin) apBegin = apEnd = pTemp;
    else {
        apEnd->apNext = pTemp;
        apEnd = pTemp;
    }
}

void PersonList::ViewList() {
    int i = 0;
    for (Node* pCur = apBegin; pCur; pCur = pCur->apNext) {
        cout << ++i << " ) ";
        pCur->apElem->View();
    }
}

void Persona::Push() {
    PersonList::PushEnd(this);
}

```

5. Реализация конструкторов с параметрами и деструктора.

- Для класса Person:

```
class PersonList {
class Node {
public:
    Node(Persona*);
    Node();
    Node* apNext;
    Persona* apElem;
};
static Node* apBegin;
static Node* apEnd;
public:
    static void PushEnd(Persona*);
    static void ViewList();
};
```

- Для класса HeadofDed:

```
class HeadOfDed :
public Persona
{
protected:
    string aNameOfDep;
public:
    HeadOfDed();
    HeadOfDed(const HeadOfDed&);
    HeadOfDed(int, string, string);
    ~HeadOfDed();
    void View() override;
};
```

- Для класса Prepod:

```
class Prepod :
public Persona
{
protected:
    string aAcademicSubject;
public:
    Prepod();
    Prepod(const Prepod&);
    Prepod(int, string, string);
    ~Prepod();
    void View() override;
};
```

- Для класса Student:

```
class Student :
public Persona {
protected:
    int aCourse;
    string aFaculty;
public:
    Student();
    Student(const Student&);
    Student(int, string, string, int);
    void View() override;
};
```

6. Реализация методов для добавления объектов в список.

```
class PersonList {
    class Node {
    public:
        Node(Persona*);
        Node();
        Node* apNext;
        Persona* apElem;
    };
    static Node* apBegin;
    static Node* apEnd;
public:
    static void PushEnd(Persona*);
    static void ViewList();
};

PersonList::Node* PersonList::apBegin = nullptr;
PersonList::Node* PersonList::apEnd = nullptr;
PersonList::Node::Node() : apNext(nullptr) {}
PersonList::Node::Node(Persona* elem) : apNext(nullptr), apElem(elem) {}
void PersonList::PushEnd(Persona* elem) {
    Node* pTemp = new Node(elem);
    if (!apBegin) apBegin = apEnd = pTemp;
    else {
        apEnd->apNext = pTemp;
        apEnd = pTemp;
    }
}
}}
```

7. Реализация метода для просмотра списка.

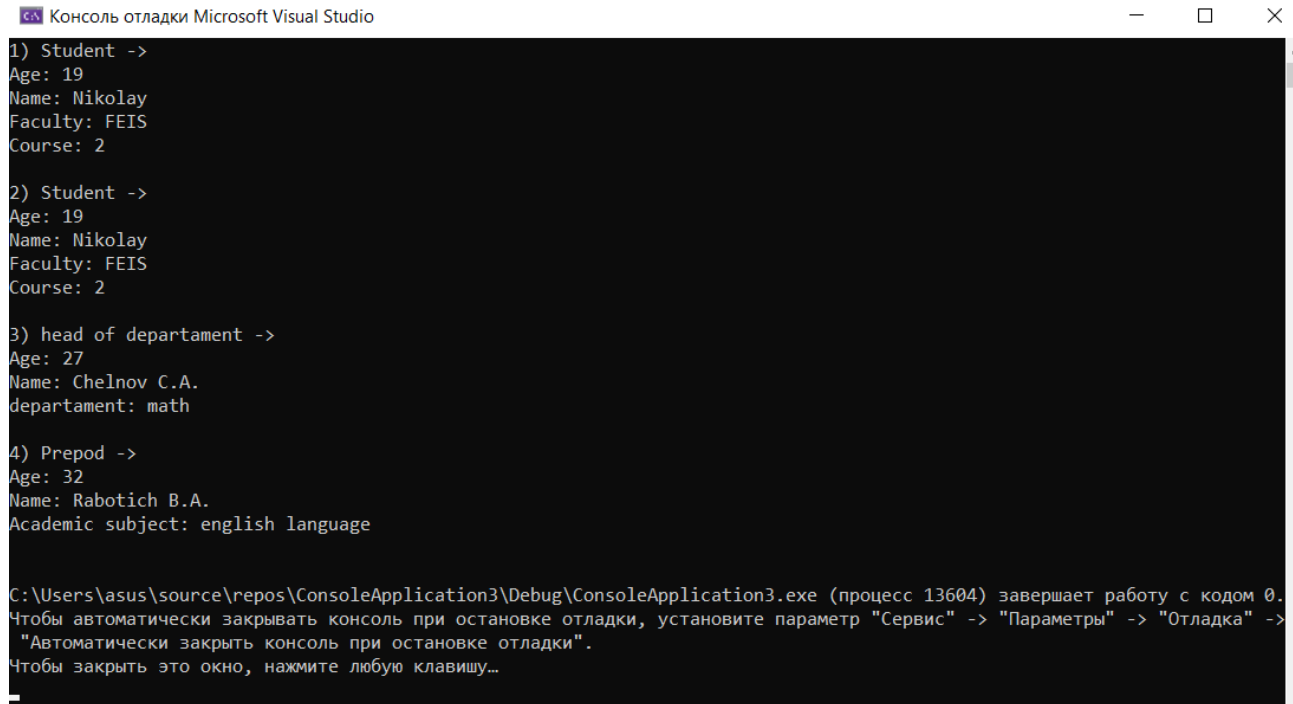
```
void PersonList::ViewList() {
    int i = 0;
    for (Node* pCur = apBegin; pCur; pCur = pCur->apNext) {
        cout << ++i << " " << pCur->apElem->View();
    }
}
```

8. Листинг демонстрационной программы.

```
#include <iostream>
#include "Persona.h"
#include "HeadOfDed.h"
#include "Prepod.h"
#include "Student.h"

int main() {
    Student a(19, "Nikolay", "FEIS", 2);
    Student b(a);
    HeadOfDed c(27, "Chelnov C.A.", "math");
    Prepod d(32, "Rabotich B.A.", "english language");
    a.Push();
    b.Push();
    c.Push();
    d.Push();
    PersonList::ViewList();
}
```

Результат выполнения:



```
Консоль отладки Microsoft Visual Studio

1) Student ->
Age: 19
Name: Nikolay
Faculty: FEIS
Course: 2

2) Student ->
Age: 19
Name: Nikolay
Faculty: FEIS
Course: 2

3) head of departament ->
Age: 27
Name: Chelnov C.A.
departament: math

4) Prepod ->
Age: 32
Name: Rabotich B.A.
Academic subject: english language

C:\Users\asus\source\repos\ConsoleApplication3\Debug\ConsoleApplication3.exe (процесс 13604) завершает работу с кодом 0.
Чтобы автоматически закрывать консоль при остановке отладки, установите параметр "Сервис" -> "Параметры" -> "Отладка" ->
"Автоматически закрыть консоль при остановке отладки".
Чтобы закрыть это окно, нажмите любую клавишу...
```

9. Объяснение необходимости виртуальных функций. Следует показать, какие результаты будут в случае виртуальных и не виртуальных функций.

При наследовании бывает необходимо, чтобы поведение некоторых методов базового класса и классов-наследников различалось, именно для этого и требуется наличие виртуальных функций `virtual void View(); virtual void Push();` В моем случае, при отсутствии виртуальной функции нельзя будет показать конечный список, корректно напечатать информацию и добавить объект в список.

10. Вывод:

Получил практические навыки реализации классов на C++.