МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
УЧРЕЖДЕНИЯ ОБРАЗОВВАНИЯ
«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
Кафедра интеллектуальных информационных технологий

Отчёт по лабораторной работе №1
по дисциплине Объектно-Ориентированные Технологии Программирования и Стандарты Проектирования за I семестр
Тема: "Классы и объекты в C++"

Выполнил:
студент 2 курса
IV семестра
факультета ЭИС
группы ПО-4(1)
Галанин П. И.
«__» _____ 2021 г.

Проверил:
магистрант
кафедры ИИТ
Миндер А. В.
«__» _____ 2021 г.

Брест 2021

Отчёт по лабораторной работе №1

Тема: «Классы и объекты в C++»
Цель работы:

Ход работы:

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

Порядок выполнения работы

1. Определить иерархию классов (в соответствии с вариантом).
2. Определить в классе статическую компоненту - указатель на начало связанного списка объектов и статическую функцию для просмотра списка.
3. Реализовать классы.
4. Написать демонстрационную программу, в которой создаются объекты различных классов и помещаются в список, после чего список просматривается.
5. Сделать соответствующие методы не виртуальными и посмотреть, что будет.
6. Реализовать вариант, когда объект добавляется в список при создании, т.е. в конструкторе (смотри пункт 6 следующего раздела).

Содержание отчета

1. Титульный лист: название дисциплины; номер и наименование работы; фамилия, имя, отчество студента; дата выполнения.
2. Постановка задачи. Следует дать конкретную постановку, т.е. указать, какие классы должны быть реализованы, какие должны быть в них конструкторы, компоненты-функции и т.д.
3. Иерархия классов в виде графа.
4. Определение пользовательских классов с комментариями.
5. Реализация конструкторов с параметрами и деструктора.
6. Реализация методов для добавления объектов в список.
7. Реализация методов для просмотра списка.

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

Приложение. Варианты заданий. Перечень классов:

1. студент, преподаватель, персона, завкафедрой;
2. служащий, персона, рабочий, инженер;
3. рабочий, кадры, инженер, администрация;
4. деталь, механизм, изделие, узел;
5. организация, страховая компания, судостроительная компания, завод;
6. журнал, книга, печатное издание, учебник;
7. тест, экзамен, выпускной экзамен, испытание;
8. место, область, город, мегаполис;
9. игрушка, продукт, товар, молочный продукт;
10. квитанция, накладная, документ, чек;
11. автомобиль, поезд, транспортное средство, экспресс;
12. двигатель, двигатель внутреннего сгорания, дизель, турбореактивный двигатель;
13. республика, монархия, королевство, государство;
14. млекопитающие, парнокопытные, птицы, животное;
15. корабль, пароход, парусник, корвет.

Вариант №1

Блок-схемы классов Блок-схема наследования классов изображена на рисунке 1 (стр. 3).

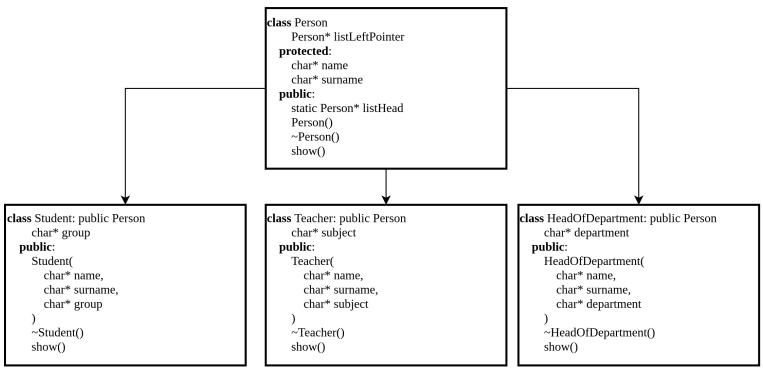

Рисунок 1 – Наследование классов

Листинг: main.hpp

```hpp
#ifndef __MAIN_HPP__
    #define __MAIN_HPP__

    #include "Person/Person.hpp"
    #include "Person/Student/Student.hpp"
    #include "Person/Teacher/Teacher.hpp"
    #include "Person/HeadOfDepartment/HeadOfDepartment.hpp"
#endif // __MAIN_HPP__
```

Листинг: main.cpp

```cpp
#include "main.hpp"

Person* Person::listHead = NULL;
int Person::listLength = 0;

int main()
{
    Person p1 = Person();

    Student s1("Daniil", "Andreychikov", "PS-4");
    Student s2("Yana", "Baiduk", "PS-4");
    Student s3("Maxim", "Borovsky", "PS-4");
    Student s4("Anastasia", "Vorobey", "PS-4");
    Student s5("Pavel", "Galanin", "PS-4");
    Student s6("Dmitry", "Gribovsky", "PS-4");
    Student s7("Alexandra", "Gritsak", "PS-4");
    Student s8("Sergey", "Eliseev", "PS-4");
    Student s9("Zhuk", "Vladislav", "PS-4");
    Student s10("Ivan", "Ivanenko", "PS-4");
    Student s11("Vladimir", "Kalinovsky", "PS-4");
    Student s12("Vladislav", "Kovalchuk", "PS-4");
    Student s13("Stanislav", "Kotashevich", "PS-4");
    Student s14("Kirill", "Krechko", "PS-4");
    Student s15("Alexey", "Kydzela", "PS-4");
    Student s16("Alexey", "Lud", "PS-4");
    Student s17("Alexander", "Mayevsky", "PS-4");
    Student s18("Nikolay", "Pivnik", "PS-4");
    Student s19("Nikita", "Prokopyuk", "PS-4");
    Student s20("Danil", "Sinyak", "PS-4");
    Student s21("Denis", "Typik", "PS-4");
    Student s22("Anastasia", "Shiba", "PS-4");
    Student s23("Vladislav", "Yuriev", "PS-4");
    Student s24("Ilya", "Yakovchik", "PS-4");

    Teacher t1("Sergey", "Anfilets", "Programming languages");
    Teacher t2("Maria", "Khatskevich", "Programming languages");
    Teacher t3("Tatiana", "Glushchenko", "Discrete Math");
    Teacher t4("Ivan", "Gladki", "Hight Math");
    Teacher t5("Oksana", "Voitsekhovich", "Interface and software development technologies");
    Teacher t6("Alexander", "Kroshchenko", "Decision making methods and algorithms");

    HeadOfDepartment h1("Vladimir", "Golovko", "Intelligent information technology");

    p1.printList();

    return 0;
}
```

Листинг: Person.hpp

```
#ifndef __PERSON_HPP__
    #define __PERSON_HPP__

    #include <iostream>
    #include <cstring>
    using namespace std;

    class Person
    {
        private:
            Person* listPointerLeft;
        protected:
            char* name;
            char* surname;
        public:
            static Person* listHead;
            static int listLength;

            Person();
            Person(char* name, char* surname);
            ~Person();
            void printList();
            // Если убрать virtual, то функция, которая вызовет this->show() вызовет только эту фун
    кцию,
            // Если слово virtual оставить, то если в наследованом классе есть такой метод show(),
            // то сработает именно новый метод show(), а не этот
            virtual void show();
    };
#endif // __PERSON_HPP__
```

Листинг: Person.cpp

```
#include "Person.hpp"

Person::Person()
{
    this->listPointerLeft = listHead;
    listHead = this;
    listLength += 1;

    this->name = new char[1];
    strcpy(this->name, "_");

    this->surname = new char[1];
    strcpy(this->surname, "_");
}

Person::Person(char* name, char* surname)
{
    this->listPointerLeft = listHead;
    listHead = this;
    listLength += 1;

    this->name = new char[strlen(name)];
    strcpy(this->name, name);

    this->surname = new char[strlen(surname)];
    strcpy(this->surname, surname);
```

```
27  }
28
29  Person::~Person()
30  {
31      delete this->name;
32      delete this->surname;
33  }
34
35  void Person::printList()
36  {
37      cout << endl;
38      cout << "[" << endl;
39      for (Person* temp = listHead; temp != NULL; temp = temp->listPointerLeft)
40      {
41          temp->show();
42      }
43      cout << "]" << endl;
44      cout << endl;
45  }
46
47  void Person::show()
48  {
49      cout << "\t{" << endl
50          << "\t\t" << "\"name\": \"" << this->name << "\"," << endl
51          << "\t\t" << "\"surname\": \"" << this->surname << "\"," << endl
52      << "\t},"  << endl;
53  }
```

Листинг: Student.hpp

```
#ifndef __STUDENT_HPP__
    #define __STUDENT_HPP__

    #include <iostream>
    #include <cstring>
    using namespace std;

    #include "../Person.hpp"

    class Student: public Person
    {
        private:
            char* group;
        public:
            Student(const char* name, const char* surname, const char* group);
            ~Student();
            void show();
    };
#endif // __STUDENT_HPP__
```

Листинг: Student.cpp

```
#include "Student.hpp"

Student::Student(const char* name, const char* surname, const char* group)
{
    this->name = new char[strlen(name)];
    strcpy(this->name, name);

    this->surname = new char[strlen(surname)];
    strcpy(this->surname, surname);

    this->group = new char[strlen(group)];
    strcpy(this->group, group);
}

Student::~Student()
{
    delete this->group;
}

void Student::show()
{
    cout << "\t{" << endl
        << "\t\t" << "\"name\": \"" << this->name << "\"," << endl
        << "\t\t" << "\"surname\": \"" << this->surname << "\"," << endl
        << "\t\t" << "\"group\": \"" << this->group << "\"," << endl
    << "\t},"  << endl;
}
```

Листинг: Teacher.hpp

```hpp
#ifndef __TEACHER_HPP__
    #define __TEACHER_HPP__

    #include <iostream>
    #include <cstring>
    using namespace std;

    #include "../Person.hpp"

    class Teacher: public Person
    {
        private:
            char* subject;
        public:
            Teacher(const char* name, const char* surname, const char* subject);
            ~Teacher();
            void show();
    };
#endif // __TEACHER_HPP__
```

Листинг: Teacher.cpp

```cpp
#include "Teacher.hpp"

Teacher::Teacher(const char* name, const char* surname, const char* subject)
{
    this->name = new char[strlen(name)];
    strcpy(this->name, name);

    this->surname = new char[strlen(surname)];
    strcpy(this->surname, surname);

    this->subject = new char[strlen(subject)];
    strcpy(this->subject, subject);
}

Teacher::~Teacher()
{
    delete this->subject;
}

void Teacher::show()
{
    cout << "\t{" << endl
        << "\t\t" << "\"name\": \"" << this->name << "\"," << endl
        << "\t\t" << "\"surname\": \"" << this->surname << "\"," << endl
        << "\t\t" << "\"subject\": \"" << this->subject << "\"," << endl
    << "\t}," << endl;
}
```

Листинг: HeadOfDepartment.hpp

```hpp
#ifndef __HEADOFDEPARTMENT_HPP__
    #define __HEADOFDEPARTMENT_HPP__

    #include <iostream>
    #include <cstring>
    using namespace std;

    #include "../Person.hpp"

    class HeadOfDepartment: public Person
    {
        private:
            char* department;
        public:
            HeadOfDepartment(const char* name, const char* surname, const char* department);
            ~HeadOfDepartment();
            void show();
    };
#endif // __HEADOFDEPARTMENT_HPP__
```

Листинг: HeadOfDepartment.cpp

```cpp
#include "HeadOfDepartment.hpp"

HeadOfDepartment::HeadOfDepartment(const char* name, const char* surname, const char* department)
{
    this->name = new char[strlen(name)];
    strcpy(this->name, name);

    this->surname = new char[strlen(surname)];
    strcpy(this->surname, surname);

    this->department = new char[strlen(department)];
    strcpy(this->department, department);
}

HeadOfDepartment::~HeadOfDepartment()
{
    delete this->department;
}

void HeadOfDepartment::show()
{
    cout << "\t{" << endl
        << "\t\t" << "\"name\": \"" << this->name << "\"," << endl
        << "\t\t" << "\"surname\": \"" << this->surname << "\"," << endl
        << "\t\t" << "\"department\": \"" << this->department << "\"," << endl
    << "\t}," << endl;
}
```

Листинг: Out

```
[
    {
        "name": "Vladimir",
        "surname": "Golovko",
        "department": "Intelligent information technology",
    },
    {
        "name": "Alexander",
        "surname": "Kroshchenko",
        "subject": "Decision making methods and algorithms",
    },
    {
        "name": "Oksana",
        "surname": "Voitsekhovich",
        "subject": "Interface and software development technologies",
    },
    {
        "name": "Ivan",
        "surname": "Gladki",
        "subject": "Hight Math",
    },
    {
        "name": "Tatiana",
        "surname": "Glushchenko",
        "subject": "Discrete Math",
    },
    {
        "name": "Maria",
        "surname": "Khatskevich",
        "subject": "Programming languages",
    },
    {
        "name": "Sergey",
        "surname": "Anfilets",
        "subject": "Programming languages",
    },
    {
        "name": "Ilya",
        "surname": "Yakovchik",
        "group": "PS-4",
    },
    {
        "name": "Vladislav",
        "surname": "Yuriev",
        "group": "PS-4",
    },
    {
        "name": "Anastasia",
        "surname": "Shiba",
        "group": "PS-4",
    },
    {
        "name": "Denis",
        "surname": "Typik",
        "group": "PS-4",
    },
    {
        "name": "Danil",
```

```
            "surname": "Sinyak",
            "group": "PS-4",
        },
        {
            "name": "Nikita",
            "surname": "Prokopyuk",
            "group": "PS-4",
        },
        {
            "name": "Nikolay",
            "surname": "Pivnik",
            "group": "PS-4",
        },
        {
            "name": "Alexander",
            "surname": "Mayevsky",
            "group": "PS-4",
        },
        {
            "name": "Alexey",
            "surname": "Lud",
            "group": "PS-4",
        },
        {
            "name": "Alexey",
            "surname": "Kydzela",
            "group": "PS-4",
        },
        {
            "name": "Kirill",
            "surname": "Krechko",
            "group": "PS-4",
        },
        {
            "name": "Stanislav",
            "surname": "Kotashevich",
            "group": "PS-4",
        },
        {
            "name": "Vladislav",
            "surname": "Kovalchuk",
            "group": "PS-4",
        },
        {
            "name": "Vladimir",
            "surname": "Kalinovsky",
            "group": "PS-4",
        },
        {
            "name": "Ivan",
            "surname": "Ivanenko",
            "group": "PS-4",
        },
        {
            "name": "Zhuk",
            "surname": "Vladislav",
            "group": "PS-4",
        },
        {
            "name": "Sergey",
```

```
        "surname": "Eliseev",
        "group": "PS-4",
    },
    {
        "name": "Alexandra",
        "surname": "Gritsak",
        "group": "PS-4",
    },
    {
        "name": "Dmitry",
        "surname": "Gribovsky",
        "group": "PS-4",
    },
    {
        "name": "Pavel",
        "surname": "Galanin",
        "group": "PS-4",
    },
    {
        "name": "Anastasia",
        "surname": "Vorobey",
        "group": "PS-4",
    },
    {
        "name": "Maxim",
        "surname": "Borovsky",
        "group": "PS-4",
    },
    {
        "name": "Yana",
        "surname": "Baiduk",
        "group": "PS-4",
    },
    {
        "name": "Daniil",
        "surname": "Andreychikov",
        "group": "PS-4",
    },
    {
        "name": "_",
        "surname": "_",
    },
]
```

Вывод: Научились делать наследование. Реализовали три класса (Student, Teacher, HeadOf Department), которые наследую один класс (Person). Для троих классов (Student, Teacher, HeadOfDepartment) переопределили метод show, но чтобы он переопределялся, то использовали virtual в базовом классе (Если в базовом классе вызывается show, а это другой класс, который просто наследует этот, то чтобы не вызывался базовый метод прописали виртуал). Для базового класса (Person) реализовали динамическую структура данных «Список», используя статический указатель на голову (чтобы во всех классах была известна одна голова списка), статический размер списка (чтобы во всех класса была информация о размере списка).

Список использованных источников

1. Наследование в ООП пример. Что такое наследование. Для чего нужно наследование классов. ООП. C++ #98
   https://www.youtube.com/watch?v=O7ruEWCa7zc
2. Модификаторы доступа при наследовании. private public protected Спецификаторы доступа. ООП. C++ #99
   https://www.youtube.com/watch?v=6udKffus77A
3. Перегрузка операторов пример. ООП. Перегрузка оператора присваивания. C++ Для начинающих. Урок#83
   https://www.youtube.com/watch?v=nMM98LVJn-U
4. Перегрузка оператора равенства == и не равно !=. Перегрузка логических операторов сравнения. C++ #84
   https://www.youtube.com/watch?v=UsezbK-3BL0
5. Дружественные функции и классы пример. Для чего используются. Как определяются. Для двух классов #88
   https://www.youtube.com/watch?v=Ic19I0kcBnU
6. Дружественный метод класса. ООП. friend c++ что это. Функции друзья. C++ Для начинающих. Урок #90
   https://www.youtube.com/watch?v=c3FJv4v7NlU
7. Знания полученые от лабораторной работы №4 "Стэки и очереди"по дисциплине Алгоритмы и Структуры Данных (БрГТУ ПОИТ)
8. Перегрузка оператора индексирования. Перегрузка операторов пример. C++ Для начинающих. Урок #87
   https://www.youtube.com/watch?v=f-N4QsyLluM
9. Виртуальные методы класса c++. Ключевое слово virtual. Ключевое слово override. ООП. C++ #103
   https://www.youtube.com/watch?v=YlbFPAugFNA