ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ "САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М. А. БОНЧ-БРУЕВИЧА"

Факультет инфокоммуникационных сетей и систем
Кафедра программной инженерии и вычислительной техники

ЛАБОРАТОРНАЯ РАБОТА №4

«КЛАССЫ»

по дисциплине «ООП»

Вариант №13

Выполнил студент группы
ИКПИ-12
Музычук Артем

Санкт-Петербург
2022

1. Постановка задачи

Дополнить систему, состоящую из трех классов COne, CTwo и CThree, которые были разработаны в лабораторной работе 3, новым классом CFour. Новый класс должен быть связан **public** наследованием с классом CThree. Класс CFour должен иметь одно поля, которое выбирается студентом самостоятельно. Для разрабатываемого класса написать конструкторы умолчания, с параметрами и конструктор копирования, деструктор, методы доступа и метод print(). Метод print() в классах CTwo, CThree и CFour должен быть виртуальным. Написать тестовую программу для проверки работоспособности разработанных классов. Разработать глобальную функцию printAll(), имеющую два параметра: массив указателей типа CTwo* и количество элементов в этом массиве **int** n.

В тестовой программе массив указателей должен быть инициализирован адресами объектов типа CTwo, CThree и CFour.

2. Таблицы атрибутов классов

Таблица атрибутов класса COne

N	Назначение	Идентификатор	Секция
1	Число типа long	L (long)	protected
2	Строка типа string	S (string)	protected

Таблица атрибутов класса CTwo

N	Назначение	Идентификатор	Секция
1	Указатель на объект типа COne	P (COne *)	protected
2	Строка типа string	S (string)	protected

Таблица атрибутов класса CThree

N	Назначение	Идентификатор	Секция
1	Указатель на объект типа COne	P (COne *)	protected
2	Строка типа string	S (string)	protected

3	Указатель на массив	a (char *)	protected

Таблица атрибутов класса CFour

N	Назначение	Идентификатор	Секция
1	Число типа int (Пол)	Int pol	protected

3. Описание методов класса Cone

COne() – конструктор по умолочанию

COne(string input , long L) : S(move(input)), L(L) {} – конутруктор с агрументами

COne(const COne &arg) – конструктор копирования

COne &operator=(const COne &arg) – оператор копирования

const long getValue() – взятие значения

const string getString() – взятие строки

size_t len() – задание длины

void print() -вывод на экран

~COne() – диструктор

4. Описание методов класса Ctwo

CTwo() – конструктор по умолочанию

CTwo(string _s, string _ps, int _number) – конструктор с параметрами

CTwo(const CTwo &arg) – конструктор копирования

CTwo &operator=(const CTwo &arg) - оператор копирования

const COne *getCOne() – обратиться к классу Cone

const string getString() – взятие строки

size_t len() – задание длины

virtual void print() – вывод на экран

~CTwo() – диструктор

5. Описание методов класса Cthree

CThree():CTwo() – конструктор по умолочанию

CThree(string s, string ps, int number) : CTwo(move(s), move(ps), number) – конструктор с параметрами

CThree(const CThree &arg) : CThree(arg.S, arg.P->S, arg.P->L) – конструктор копирования

CThree &operator=(const CThree &arg) – оператор копирования

char &operator[](size_t idx) – оператор копирования

void print() const override – вывод на экран

~CThree() – диструктор

6. Описание методов класса Cfour

CFour():CThree() – конструктор по умолочанию

CFour(string s, string ps, string cps, int number, int pol) : CThree(move(s), move(ps), number) - с параметрами

CFour(const CFour &arg) : CThree(arg) – конструктор копирования

void setPol(int pol) – установить пол

float getPol() – взять пол

CFour &operator=(const CFour &arg) – оператор копирования

void print() const override – вывод на экран

~CFour() – диструктор

7. Описание вывод и теста

void printAll(CTwo **t, size_t n) – вывести все
void test() – сделать тест и вывести результат на экран

8. Код программы

```
#include <iostream>
#include <string>

using namespace std;

class COne {
   public:
      long L;
      string S;

   public:
```

```
    COne(){
        this->L = 0;
        this->S = "0";
    };

    COne(string input , long L) : S(move(input)), L(L) {}

    COne(const COne &arg) {
        L = arg.L;
        S = arg.S;
    }

    COne &operator=(const COne &arg) {
        COne temp(arg);
        swap(L, temp.L);
        swap(S, temp.S);
        return *this;
    }

    ~COne() {
        S.clear();
    };

    long getValue() const {
        return L;
    }

    string getString() const
    {
        return S;
    }

    size_t len() const {
        return S.size();
    }

    void print() const {
        cout << "\nCOne: " << L << ", "<< S ;
    }

    friend class CTwo;
};

class CTwo {
    protected:
        string S;
        COne *P; // ОТНОШЕНИЕ ВКЛЮЧЕНИЯ

    public:

    CTwo(){
        S = "0";
        P = new COne();
    }

    CTwo(string _s, string _ps, int _number) {
        S=_s;
```

```
        P = new COne(move(_ps), _number);
    }

    CTwo(const CTwo &arg) {
        P = new COne(*arg.P);
        S = arg.S;
    }

    CTwo &operator=(const CTwo &arg) {
        CTwo temp(arg);
        swap(P, temp.P);
        swap(S, temp.S);
        return *this;
    }

    ~CTwo() {
        delete P;
        S.clear();
    }

    COne *getCOne() const {
        return P;
    }

    string getString() const {
        return S;
    }

    size_t len() const {
        return S.size();
    }

    virtual void print() const {
        cout <<"CTwo: ";
        if (P) {
            cout << '\"' << S << "\", ";
            P->print();
        } else { cout << "undefined"; }
    }
};

class CThree : public CTwo { // ОТНОШЕНИЕ НАСЛЕДОВАНИЯ
    public:
        char *a;

    public:

    CThree():CTwo(){
        a = "0";
    }

    CThree(string s, string ps, int number) : CTwo(move(s), move(ps), number) {
        if (number > 0) {
            a = new char[number] {'8','9','6','2','1','2','1','6','0','9','0'};
        } else { cout << "undefined"; }
    }
```

```
    CThree(const CThree &arg) : CThree(arg.S, arg.P->S, arg.P->L) {
        std::copy(arg.a, arg.a + P->L, a);
    }

    CThree &operator=(const CThree &arg) {
        CThree temp(arg);
        swap(*this, temp);
        return *this;
    }

    char &operator[](size_t idx) {
        return a[idx];
    }

    void print() const override{
        cout << "CThree: ";
            cout << "[";
            for (int i = 0; i < 11; i++) {
                cout << a[i];
            }
            cout << "]\n";
        CTwo::print();
    }
};

class CFour : public CThree { // ОТНОШЕНИЕ НАСЛЕДОВАНИЯ
    protected:
        int pol;

    public:

    CFour():CThree(){
        pol = 10;
    }

    CFour(string s, string ps, string cps, int number, int pol) : CThree(move(s), move(ps), number) {
        this->pol = pol;
    }

    CFour(const CFour &arg) : CThree(arg) {
        this->pol = arg.pol;
    }

   void setPol(int pol){
        this-> pol= pol;
    }

    float getPol(){
        return this->pol;
    }

    CFour &operator=(const CFour &arg) {
        CFour temp(arg);
        swap(*this, temp);
        return *this;
    }
```

```
    void print() const override {
        cout << "CFour: " << pol << "\n";

        CThree::print();
    }

    ~CFour(){}
};

void printAll(CTwo **t, size_t n) {
    for (size_t i = 0; i < n; ++i) {
        (t[i]->print()), cout << endl;
    }
}

void test() {
    CTwo *Array[3];
    Array[0] = new CTwo("Артем", "Музычук", 19);
    Array[1] = new CThree("Артем", "Музычук", 19);
    Array[2] = new CFour("Артем", "Музычук", "89621216090", 19, 1);
    printAll(Array, 3);
    delete Array[0];
    delete Array[1];
    delete Array[2];
}

int main() {
    int n,N=11;
    int pol = 0;
    string *s1= new string;
    string *s2= new string;
    string *cps= new string;


    cout<<"Введите ваши данные\n ";
    cout<<"Имя: ";
    cin>>*s1;
    cout<<"Фамилия: ";
    cin>>*s2;
    cout<<"Возраст: "; cin>>n;
    cout<<"Пол(1 или 0): "; cin>>pol;
    CFour third(*s1,*s2,*cps,n,pol);

    cout<<"\n_____\n"<<endl;
    cout<<"\t[Результат работы программы]\n"<<endl;
    cout << "\nСодержимое объекта класса\n ",third.print(),cout<< endl;
    cout<<"\n_____\n"<<endl;

    cout << "Имя и Фамилия: " << third.getString() << ' ' << third.getCOne()->getString() << endl;
    cout << "Возраст: " << third.getCOne()->getValue() << endl;
    cout << "Телефон: " << third.a << endl;
    cout << "Пол: " << third.getPol() << endl;


    // Копирование конструктором
    cout << "\n[Осуществив копирование с помощью конструктора, создадим новый объект класса]\n" << endl;
    CFour forth(third);
```

```
    cout << "third: ", third.print(), cout << endl <<endl;
    cout << "forth: ", forth.print(), cout << endl;

        // Тест
        cout << "\n PRINT ALL:\n";
        test();

    return 0;
}
```

9. Тесты:

//Введенные значения

```
Введите ваши данные
 Имя: артем
Фамилия: музычук
Возраст: 19
Пол(1 или 0): 1
```

//Результат работы программы

```
_____________________

        [Результат работы программы]


Содержимое объекта класса
 CFour: 1
CThree: [89621216090]
CTwo: "артем",
COne: 19, музычук


_____________________

Имя и Фамилия: артем музычук
Возраст: 19
Телефон: 89621216090
Пол: 1

[Осуществив копирование с помощью конструктора, создадим новый объект класса]

third: CFour: 1
CThree: [89621216090]
CTwo: "артем",
COne: 19, музычук

forth: CFour: 1
CThree: [89621216090]
CTwo: "артем",
COne: 19, музычук

 PRINT ALL:
CTwo: "Артем",
COne: 19, Музычук
CThree: [89621216090]
CTwo: "Артем",
COne: 19, Музычук
CFour: 1
CThree: [89621216090]
CTwo: "Артем",
COne: 19, Музычук
```

10. Выводы

В ходе лабораторной работы, мы разработали класс CFour, связанный наследованием с классом CThree. Разработаны вирутальные функции print() для CTwo, CThree и CFour.. Разработана глобальная функция printAll(), имеющая два параметра: массив указателей типа CTwo* и количество элементов в этом массиве int n. Также написаны тесты, которые показали, что наследование реализовано верно.