Правительство Российской Федерации

Федеральное государственное автономное образовательное учреждение
высшего образования «Национальный исследовательский университет
«Высшая школа экономики»

Кафедра «Компьютерная безопасность»

ОТЧЕТ
К ЛАБОРАТОРНОЙ РАБОТЕ №6

по дисциплине

«Языки программирования»

Работу выполнила
студентка группы
СКБ-232 _____ Д.В. Иванова
подпись, дата

Работу проверил _____ С.А. Булгаков
подпись, дата

Москва 2023

Содержание

Постановка задачи

Нужно было разработать реализовать метод класса Town: ‘Town ReadXML(std::istream)‘ считывающий представление объекта класса на языке XML из потока ввода языка Си++. Предусмотреть возможность поступления некорректных данных. Для обработки нештатных ситуаций задействовать механизм исключений.

1 Описание функции main

Описание класса

Класс Town содержит 4 приватных поля: количество жителей, имя мэра, название города и название страны. У класса три конструктора: без параметров (который по умолчанию присваивает страну Russia, город Moscow, мэра Sobyanin и количество жителей 1000000), с 3 параметрами: название города, имя мэра и количество жителей и конструктор копирования. А также деструктор. Кроме того, класс содержит методы: изменение майора, изменение количества жителей, возвращение количества жителей, имя мэра, название города, название страны, вывод информации о городе в формате текста и в формате XML, парсер значения тегов, парсер значений между открывающим и закрывающим тегом и метод Town ReadXML, который читает значения из xml файла и на их основе делает объект класса Town.

2 Тестирование

1 Проверка вывода в XML формате

```
Town Moscow;
Moscow.WriteXML(std::cout);
output:
<?xml version="1.0" encoding="UTF-8" ?>
    <Town>
        <NumberOfCitizens> 1000000 </NumberOfCitizens>
        <MayorName> Sobyanin </MayorName>
        <CountryName> Russia </CountryName>
        <TownName> Moscow </TownName>
    </Town>
```

2 Проверка вывода информации о городе

```
Town Moscow;
Moscow.Town_info();
output:
----------------------------------------------------------------------
Country: Russia
Town name: Moscow
Mayor: Sobyanin
Number of citizens: 1000000
----------------------------------------------------------------------
```

Город Moscow создан конструктором без параметров, поэтому подобный вывод верен.

3 Смена мэра

```
Town Moscow;
Moscow.change_mayor("C0olH4ck3r");
Moscow.Town_info();
output:
------------------------------------------------------------------
Country: Russia
Town name: Moscow
Mayor: C0olH4ck3r
Number of citizens: 1000000
------------------------------------------------------------------
```

4 Изменение числа жителей

```
Moscow.change_num_of_cit(2000000);
Moscow.Town_info();
output:
------------------------------------------------------------------
Country: Russia
Town name: Moscow
Mayor: C0olH4ck3r
Number of citizens: 2000000
------------------------------------------------------------------
```

5 Конструктор с 3 параметрами

```
Town town(5000, "kitten", "Norway", "Prehewill");
output:
------------------------------------------------------------------
Country: Norway
Town name: Prehewill
Mayor: kitten
Number of citizens: 5000
------------------------------------------------------------------
```

6 Конструктор копирования

```
Town Moscow2(Moscow);
Moscow2.Town_info();
output:
------------------------------------------------------------------
Country: Russia
Town name: Moscow
Mayor: Sobyanin
Number of citizens: 1000000
------------------------------------------------------------------
```

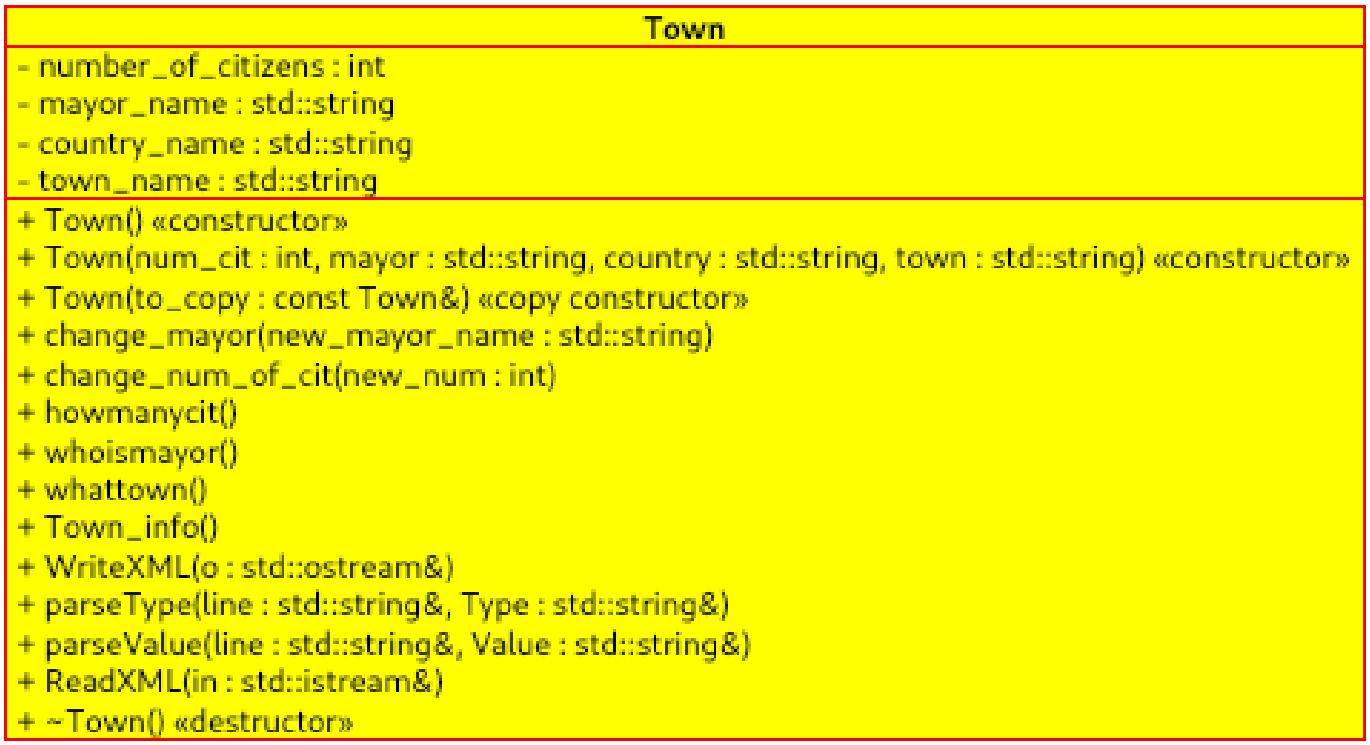
7 ReadXML

```
<?xml version="1.0" encoding="UTF-8" ?>
<Town>
     <NumberOfCitizens> 3000 </NumberOfCitizens>
     <MayorName> MayorFile </MayorName>
     <CountryName> XMLCountry </CountryName>
     <TownName> TownCorrect </TownName>
</Town>
```

```
Moscow = Moscow.ReadXML(inputFile);
Moscow.Town_info();
----------------------------------------------------------------------
Country:  XMLCountry
Town name:  TownCorrect
Mayor:  MayorFile
Number of citizens: 3000
----------------------------------------------------------------------
```

Приложение А

UML 2.0-схема класса *Town*



Приложение Б

Код Town.h

```
#ifndef  Town_h
#define  Town_h

#include <string>

class Town{
    private:
        int number_of_citizens;
        std::string mayor_name;
        std::string country_name;
        std::string town_name;

    public:
        Town();

        Town(int num_cit, std::string mayor, std::string country, std::string town);

        Town(const Town &to_copy);

        void change_mayor(std::string new_mayor_name);

        void change_num_of_cit(int new_num);

        int howmanycit();

        std::string whoismayor();

        std::string whatcountry();

        std::string whattown();

        void Town_info();

        void WriteXML(std::ostream& o);

        ~Town();
};

#endif //Town_h
```

Код Town.cpp

```
#include "Town.h"
#include <iostream>
#include <cstdlib>
#include <string>
#include <exception>

Town::Town(){ //constructor
    number_of_citizens = 1000000;
    mayor_name = "Sobyanin";
    country_name = "Russia";
    town_name = "Moscow";
}
Town::Town(int num_cit, std::string mayor, std::string country, std::string town) //second constructor
    : number_of_citizens(num_cit), mayor_name(mayor), country_name(country), town_name(town) {}

Town::Town(const Town &to_copy){ //copy constructor
    if (to_copy.town_name != "") town_name = to_copy.town_name;
    if (to_copy.mayor_name != "") mayor_name = to_copy.mayor_name;
    if (to_copy.country_name != "") country_name = to_copy.country_name;
    number_of_citizens = to_copy.number_of_citizens;
}
void Town::change_mayor(std::string new_mayor_name){
    this -> mayor_name = new_mayor_name;
}
void Town::change_num_of_cit(int new_num){
    if (new_num<0)
        throw "Number of people can't be negative\n";
    this -> number_of_citizens = new_num;
}

void Town::change_country_name(std::string& new_country){
    this -> country_name = new_country;
}
```

```
void Town::change_town_name(std::string& new_town){
    this -> town_name = new_town;
}

int Town::howmanycit() { return number_of_citizens; }

std::string& Town::whoismayor() { return mayor_name; }

std::string& Town::whatcountry() { return country_name; }

std::string& Town::whattown() { return town_name; }

void Town::Town_info(){
    std::cout << "-----------------------------------------------------------------------\n";
    std::cout << "Country: " << country_name << "\n";
    std::cout << "Town name: " << town_name << "\n";
    std::cout << "Mayor: " << mayor_name << "\n";
    std::cout << "Number of citizens: " << number_of_citizens << "\n";
    std::cout << "-----------------------------------------------------------------------\n";
}

void Town::WriteXML(std::ostream& o){
    o << "<?xml version=\"1.0\" encoding=\"UTF-8\" ?>" << std::endl;
    o << "    <Town> " << std::endl;
    o << "        <NumberOfCitizens> " << howmanycit() << " </NumberOfCitizens>" << std::endl;
    o << "        <MayorName> " << whoismayor() << " </MayorName>" << std::endl;
    o << "        <CountryName> " << whatcountry() << " </CountryName>" << std::endl;
    o << "        <TownName> " << whattown() << " </TownName>" << std::endl;
    o << "    </Town> " << std::endl;
}

std::string& Town::parseType(std::string& line, std::string& Type){
    Type ="";
    std::string endType = "";
    bool inOpNm = false, inEndNm = false, OpSeen = false;
    for (int i=0; i<line.size(); i++){
        if (line[i] == '<' && !OpSeen)
            inOpNm = true;
        else if (line[i] == '/')
            inEndNm = true;
        else if (line[i]!='>' && inOpNm)
            Type.push_back(line[i]);
        else if (line[i]!='>' && inEndNm)
            endType.push_back(line[i]);
        else if (line[i] == '>'){
            inOpNm = false;
            OpSeen = true;
            inEndNm = false;
        }
    }
    if (Type != endType)
        throw "Different open and close tags\n";
    return Type;
}

std::string& Town::parseValue(std::string& line, std::string& Value){
    Value = "";
    bool Element = false;
    for (int i=0; i<line.size(); i++){
        if (line[i] == '>' && !Element)
            Element = true;
        else if (Element && line[i] != '<')
            Value.push_back(line[i]);
        else if (line[i] == '<')
            Element = false;
    }
    return Value;
}

int stringToInt(const std::string& str) {
    int result = 0;
    bool inNum = false;
    for (size_t i = 0; i < str.size(); i++) {
        if ( str[i]>=48 && str[i]<=57){
            result = (result * 10) + (str[i] - '0');
            inNum =true;
        }
        else if (str[i]==' ' && inNum==true)
            inNum=false;
```

```
        else if (str[i]!=' ' && inNum==true)
            throw "Something wrong with number\n";
    }
    return result;
}

Town Town::ReadXML(std::istream& in){
    Town sometown;
    if (!in)
        throw "Can't open file\n";
    if( in.get() == '<' ){
        if( in.get() == '?' ){
            std::string str;
            in >> str;
            if( str == "xml" ){
                in >> str;
                if( str == "version=\"1.0\"" ){
                    in >> str;
                    if( str == "encoding=\"UTF-8\"" ){
                        in >> str;
                        if( str == "?>" ){
                            std::cout << "xml file is valid;)" << std::endl;
                            std::string line, Nm, Vl;
                            bool insideElement = false, selfClosingTag = false;
                            int cit =0;
                            while (std::getline(in, line)){
                                if (line.find("</Town>") != std::string::npos || line.find("<Town>") != std::string
                                    continue;
                                std::string& link = line;
                                if (parseType(link, str) == "NumberOfCitizens"){
                                    cit = stringToInt(parseValue(link, str));
                                    sometown.change_num_of_cit(cit);
                                }
                                if (parseType(link, str) == "MayorName")
                                    sometown.change_mayor(parseValue(link, str));
                                if (parseType(link, str) == "CountryName")
                                    sometown.change_country_name(parseValue(link, str));
                                if (parseType(link, str) == "TownName")
                                    sometown.change_town_name(parseValue(link, str));
                            }
                        }
                    }
                }
            }
        }
    }
    return sometown;

}

Town::~Town(){} //destrusctor
```

Код Classes.cpp

```
#include <iostream>
#include <fstream>
#include <string>
#include "Town.h"
#include "Terminate.h"

//g++ .\Classes.cpp -std=c++98 -pedantic -o .\Classes.exe
int main(){

    try
    {
        Town Moscow, town(5000, "kitten", "Norway", "Prehewill");
        std::ifstream inputFile("file.xml");
        Moscow.WriteXML(std::cout);
        Moscow = Moscow.ReadXML(inputFile);
        std::cout << "here we are\n";
        Moscow.Town_info();
    }
    catch (const char* ex)
    {
        std::cerr << "Caught exception: " << ex << std::endl;
    }
}
```