**暨南大学本科实验报告专用纸**

课程名称 C++程序设计 成绩评定

实验项目名称 共享智能指针编程实验 指导教师 王勇杰

实验项目编号 4 实验项目类型 编程型 实验地点 C105

学生姓名 李炜鹏 学号 2017052544

学院 智能科学与工程学院 系 计算机 专业 信息安全

实验时间2019年12月10日 午～12月13日 午 温度 ℃湿度

1. **实验目的**
2. 熟悉类的设计过程
3. 熟悉共享内存指针的概念以及使用（声明、初始化、取值等）
4. 实验要求：

阅读课本第12章，完成练习12.2：编写你自己的StrBlob类，包含const版本的front和back。

1. 实验报告要求：
   1. 实验目的
   2. 实验原理
   3. 在程序中适当地作注释
   4. 实验结果的截图
   5. 实验评估
2. **实验原理**

本实验要求编写自己的StrBlob类，参考课本P404-P406，本代码与课本不同的是，增加了返回const类型的元素访问成员函数，再参考课本P247的内容，即可完成本次实验，现具体讨论本实验的两个主要知识点：

1. 定义使用智能指针管理内存的StrBlob类：
   1. 首先提到定义类，免不了定义数据成员（一个指向存string的vector容器的智能指针），成员函数（公有的：构造函数、size函数、empty函数、push\_back函数、pop\_back函数、元素访问函数front和back；私有的：check函数）
   2. 智能指针的内存分配与初始化：类似于普通指针的初始化方法，可以使用以下代码初始化智能指针：

shared\_ptr<string> p;

if (p && p->empty())

\*p = “hi”;

不过此方法较为麻烦，使用完指针之后还要回收指针，销毁对象、释放内存，并且可能因为忘记delete指针、使用已经释放掉的内存、或者多次释放同一块内存而造成危险，所以，我们可以使用最简洁并且最安全的方法make\_shared函数：

Shared\_ptr<string> p = make\_shared<string>(“hi”);

1. const相关知识点：
   1. 常成员函数可以运用常数据成员和非常数据成员，而非常成员函数只能够引用非常数据成员

常成员函数的声明：返回类型 常成员函数名（参数类型表） const；

非常成员函数的声明：返回类型 非常成员函数名（参数类型表）；

* 1. 返回值为const值的成员函数必定是常成员函数，因为非常成员函数没办法访问常数据成员，于是就没办法返回const值，正确的声明如下：

const 返回类型 常成员函数名（参数类型表） const；

而不可能存在这样的声明：

const 返回类型 常成员函数名（参数类型表）；

* 1. 关于const的知识点还有很多，可以参考C++ primer（第5版）P5-P58、C++面向对象程序设计（第2版）P90-P98

1. **主要仪器设备**

**仪器：**计算机

**实验环境：**Visual Studio 2017或Dev-C++

1. **源程序**

**myStrBlob.h**

#include <iostream>

#include <vector>

#include <string>

#include <memory>

using namespace std;

class StrBlob

{

public:

typedef vector<string>::size\_type size\_type;//为vector<string>::size\_type定义别名size\_type

StrBlob();//默认构造函数

StrBlob(initializer\_list<string> i1);//含参数的构造函数

size\_type size() const { return data->size(); }//返回智能指针指向的vector的大小（计算vector中含有多少个string）

bool empty() const { return data->empty(); }//判断该智能指针指向的vector是否为空

void push\_back(const string &t) { data->push\_back(t); }//增加元素

void pop\_back();//删除元素

//元素访问：目前未搞懂这几个函数的意义

string& front();

const string& front() const;//可以引用数据成员（常与非常都可以），返回常string

string& back();

const string& back() const;//可以引用数据成员（常与非常都可以），返回常string

private:

shared\_ptr<vector<string>> data;//智能指针data：类型为——类型为string的vector容器

void check(size\_type i, const string &msg) const;//如果data[i]不合法，跑出异常信息msg

};

StrBlob::StrBlob() :data(make\_shared<vector<string>>()) {}//使用make\_shared初始化智能指针

StrBlob::StrBlob(initializer\_list<string> i1):data(make\_shared<vector<string>>(i1)){}//make\_shared<数据类型>(初始化参数)

void StrBlob::check(size\_type i, const string &msg) const//常成员函数，只能引用常数据成员和非常数据成员，而不能改变他们

{

if (i >= data->size())

throw out\_of\_range(msg);

}

string& StrBlob::front()//返回智能指针所指向vector<string>当前的前一个string数据

{

check(0, "front on empty StrBlob");//如果vector为空，check会抛出一个异常

cout << this << endl;

return data->front();

}

const string& StrBlob::front() const//返回智能指针所指向vector<string>当前的前一个string数据

{

check(0, "front on empty StrBlob");//如果vector为空，check会抛出一个异常

cout << this << endl;

return data->front();

}

string& StrBlob::back()//返回智能指针所指向vector<string>当前的后一个string数据

{

check(0, "back on empty StrBlob");//如果vector为空，check会抛出一个异常

cout << this << endl;

return data->back();

}

const string& StrBlob::back() const//返回智能指针所指向vector<string>当前的后一个string数据

{

check(0, "back on empty StrBlob");//如果vector为空，check会抛出一个异常

cout << this << endl;

return data->back();

}

void StrBlob::pop\_back()//删除vector里面所有数据

{

check(0, "pop\_back on empty StrBlob");//如果vector为空，check会抛出一个异常

data->pop\_back();

}

**useMyStrBlob.cpp**

#include "myStrBlob.h"

int main()

{

initializer\_list<string> temp = { "a","ab","abc","abcd" };

StrBlob sb(temp);

cout << "sb.size: " << sb.size() << endl;

cout << "sb.empty: " << sb.empty() << endl;

string s("abcde");

sb.push\_back(s);

cout << "sb.size: " << sb.size() << endl;

sb.pop\_back();

cout << "sb.size: " << sb.size() << endl;

//元素访问函数的使用

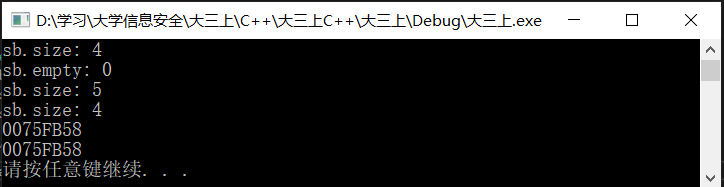
sb.front();//输出当前指向的地址

sb.back();//输出当前指向的地址

system("pause");

return 0;

}



本实验由main函数输入四个字符串a, ab, abc, abcd，并且先后调用了StrBlob类的size函数、empty函数、push\_back函数、pop\_back函数、front函数、back函数，有结果可以看出size函数输出sb对象的大小（4）、empty函数输出是否为空（0），在push\_back函数、pop\_back函数分别作用之后得到结果与预期相符，而加入了cout语句的front函数和back函数则输出当前所指向的地址。

1. **实验评估**

尽管本实验的大部分代码在课本上已经给出，不过经过上机操作之后，本实验给我带来的收获也十分大，总结如下：

* 1. 再次熟悉了类的设计，更深地理解到了C++面向对象程序设计的精髓
  2. 巩固了共享智能指针的使用（定义方法shared\_ptr、初始化方法make\_shared），理解到了共享智能指针的内存释放是在指针的作用域结束之后释放该共享指针的，而该指针指向的对象则是在没有共享智能指针指向该对象时被销毁
  3. 初步认识了C++中重要关键字const的用法（该关键字的知识点实在太多，希望在以后的学习当中完善）