ГУАП
КАФЕДРА №14

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ
ПРЕПОДАВАТЕЛЬ

Должность, уч. степень, звание подпись, дата инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №3

по курсу: ТЕХНОЛОГИЯ ПРОГРАММИРОВАНИЯ

РАБОТУ ВЫПОЛНИЛ
СТУДЕНТ ГР. 1441 М.И. Лубинец

подпись, дата инициалы, фамилия

Санкт-Петербург
2015

1. Формализация задачи

Задача №1:

Создать класс String, реализующий работу со строчками с учетом числа обращений к ним (со счетчиками). Так, например, при присвоении одной строки другой, не выделяется новая память для хранения строки, а лишь увеличивается счетчик ее использования. Класс должен предоставлять основные операции по работе со строками (сложение, сравнение, присваивание и т.д.). Данный класс должен позволять использовать себя (те данные, которые он хранит) в функциях стандартной библиотеки.

2. Листинги

Файл sharedptr.hpp

```
#ifndef SHAREDPTR_H
#define SHAREDPTR_H
#include <stdint.h>
#include <stddef.h>
#include "log.hpp"


template<typename T>
class SharedPtr {
public:
    explicit SharedPtr()
        : pptr(nullptr), ref_cnt(nullptr) {}

    explicit SharedPtr(T ptr[])
        : pptr(ptr), ref_cnt(new int32_t) { *ref_cnt = 1; }

    explicit SharedPtr(const SharedPtr& b)
        : pptr(b.pptr), ref_cnt(b.ref_cnt) { *ref_cnt++; }

   ~SharedPtr() {
        release();
    }

    void release() {
        if(!pptr && !ref_cnt) return;

        (*ref_cnt) -= 1;
        if(*ref_cnt <= 0) {
            drop();
        } else {
            pptr    = nullptr;
            ref_cnt = nullptr;
        }
    }

    void drop() {
        if(!pptr && !ref_cnt) return;

        if(pptr)    delete[] pptr;
        if(ref_cnt) delete   ref_cnt;

        pptr    = nullptr;
        ref_cnt = nullptr;
    }

    SharedPtr& operator = (const SharedPtr& b) {
        if(this == &b) return *this;

        release();

        ref_cnt = b.ref_cnt;
        pptr    = b.pptr;

        (*ref_cnt)++;

        return *this;
    }

    SharedPtr& operator = (T ptr[]) {
        set(ptr);
        return *this;
    }

    void set(T ptr[]) {
        release();

        pptr    = ptr;
        ref_cnt = new int32_t;
        (*ref_cnt) = 1;
    }
```

```
    T* get() {
        return pptr;
    }

    const T* get() const {
        return pptr;
    }

    T& operator*() {
        return *pptr;
    }

    const T& operator*() const {
        return *pptr;
    }

    T& operator[](size_t index) {
        return pptr[index];
    }

    const T& operator[](size_t index) const {
        return pptr[index];
    }

    int32_t refs() const {
        return *ref_cnt;
    }

private:
    T*         pptr;
    int32_t*   ref_cnt;
};

#endif // SHAREDPTR_H
```

Файл string.hpp

```
#ifndef STRING_H
#define STRING_H
#include <stdint.h>
#include <stddef.h>
#include "sharedptr.hpp"

namespace msvd {

using std::istream;
using std::ostream;

class String {
public:
    String();
    String(const String& b);
    String(const char* b);
    String(const SharedPtr<char>& sptr, size_t len);

    String& operator= (const String& b);
    String& operator= (const char* b);

    friend bool operator== (const String& a, const String& b);
    friend bool operator!= (const String& a, const String& b);
    friend bool operator<  (const String& a, const String& b);
    friend bool operator>  (const String& a, const String& b);
    friend bool operator<= (const String& a, const String& b);
    friend bool operator>= (const String& a, const String& b);

    friend String operator+ (const String& a, const String& b);

    char& operator[] (size_t index);
    const char& operator[] (size_t index) const;

    const char* c_str() const;

    friend ostream& operator<< (ostream& os, const String& string);
    friend istream& operator>> (istream& is, String& string);

    // Getters
    size_t   length() const { return len; }

private:
    SharedPtr<char> ptr;
    size_t          len;
};

}
#endif // STRING_H
```

Файл string.cpp

```
#include "string.hpp"
#include <cstring>

namespace msvd {

String::String() : len(0) {}

String::String(const String& b) {
    ptr = b.ptr;
    len = b.len;
}

String::String(const char* b) {
    len =  std::strlen(b) ;
    ptr.set(new char[len +1]);
    strcpy(&ptr[0], b);
}

String::String(const SharedPtr<char>& sptr, size_t len) : ptr(sptr), len(len) {}

String&
String::operator= (const String& b) {
    if(this == &b) return *this;
    ptr.release();
    ptr = b.ptr;
    len = b.len;
    return *this;
}

String&
String::operator= (const char* b) {
    len =  std::strlen(b) ;
    ptr.set(new char[len +1]);
    strcpy(&ptr[0], b);
    return *this;
}

bool operator== (const String& a, const String& b) {
    return strcmp(a.c_str(), b.c_str()) ? false : true;
}

bool operator!= (const String& a, const String& b) {
    return strcmp(a.c_str(), b.c_str()) ? true : false;
}

bool operator<  (const String& a, const String& b) {
    return strcmp(a.c_str(), b.c_str()) < 0;
}

bool operator>  (const String& a, const String& b) {
    return strcmp(a.c_str(), b.c_str()) > 0;
}

bool operator<= (const String& a, const String& b) {
    return a < b || a == b;
}

bool operator>= (const String& a, const String& b) {
    return a > b || a == b;
}


String
operator+ (const String& a, const String& b) {
    SharedPtr<char> new_str(new char[a.len + b.len + 1]);
    memcpy(&new_str[0],     &a[0], a.len);
    strcpy(&new_str[a.len], &b[0]);
    return String(new_str, a.len + b.len);
}

char&
String::operator[] (size_t index) {
    return ptr[index];
}

const char&
```

```
String::operator[] (size_t index) const {
    return ptr[index];
}




const char*
String::c_str() const {
    return &ptr[0];
}

ostream& operator<< (ostream& os, const String& string) {
    const char* pch = &string[0];
    size_t l = string.len;
    while(l--) {
        os << *pch++;
    }
    return os;
}

istream& operator>> (istream& is, String& string) {
    std::string buffer;
    is >> buffer;

    string = buffer.c_str();
    return is;
}

}
```

Файл main.cpp

```
#include <iostream>
#include <string.hpp>

using namespace std;
using namespace msvd;

int main(int argc, char *argv[]) {
    String helloworld;
    String hello2;

    {
        String hello("Hello ");
        String world("World");

        hello2 = hello;

        helloworld = hello + world;
    }

    String str1;
    cin >> str1;

    String str2 = str1;
    String str3 = str2;
    String str4 = str1;
    String str5 = str1;
    String str6 = str5;

    hello2 = str4;

    cout << hello2 << endl
         << helloworld << endl
         << str3 << endl;

    return 0;
}
```

3. Примеры

```
Вводимая строка: “Programming Technology”

[0.00000] set:            0.00000: 1
[0.00000] set:            0.00000: 1
[0.00000] operator=:      Assigning Hello  instead of 0x0
[0.00000] operator=:      Hello : 2
[0.00000] SharedPtr:      0.00000: 1
[0.00000] SharedPtr:      Hello World: 2
[0.00000] release:        Hello World: 1
[0.00000] operator=:      Assigning Hello World instead of 0x0
[0.00000] operator=:      Hello World: 2
[0.00000] release:        Hello World: 1
[0.00000] release:        World: 0
[0.00000] drop:           World: free
[0.00000] release:        Hello : 1
Programming Technology[10.99700] set:            î蜍  : 1
[10.99700] operator=:      Assigning Programming Technology  instead of 0x0
[10.99700] operator=:      Programming Technology : 2
[10.99700] operator=:      Assigning Programming Technology  instead of 0x0
[10.99700] operator=:      Programming Technology : 3
[10.99700] operator=:      Assigning Programming Technology  instead of 0x0
[10.99700] operator=:      Programming Technology : 4
[10.99700] operator=:      Assigning Programming Technology  instead of 0x0
[10.99700] operator=:      Programming Technology : 5
[10.99700] operator=:      Assigning Programming Technology  instead of 0x0
[10.99700] operator=:      Programming Technology : 6
[10.99700] release:        Hello : 0
[10.99700] drop:           Hello : free
[10.99700] operator=:      Assigning Programming Technology  instead of 0x0
[10.99700] operator=:      Programming Technology : 7
Programming Technology
Hello World
Programming Technology
[10.99700] release:        Programming Technology : 6
[10.99700] release:        Programming Technology : 5
[10.99700] release:        Programming Technology : 4
[10.99700] release:        Programming Technology : 3
[10.99700] release:        Programming Technology : 2
[10.99700] release:        Programming Technology : 1
[10.99700] release:        Programming Technology : 0
[10.99700] drop:           Programming Technology : free
[10.99700] release:        Hello World: 0
[10.99700] drop:           Hello World: free
```