**СУ „Св. Климент Охридски“, ФМИ**

Специалност „Софтуерно Инженерство“

**Обектно-ориентирано програмиране, 2019-2020 г.**

**Задача за домашно № 2**

**Спазвайте практиките за обектно-ориентирано програмиране, коментирани на упражнения и лекции.**

## **Задача 1 (2 точки)**

Имплементирайте операторите ***<<*** и **>>** за класа *MyString*

## *MyString.hpp*

| class MyString{ private:  char\* content; public:  MyString();  MyString(const char\* input);   MyString(const MyString& from);   MyString& operator=(const MyString& from);   ~MyString();   void append(char to\_append);  void print(); }; |
| --- |

*MyString.cpp*

| MyString::MyString() {  content = new char[1];  content[0] = '\0'; }  MyString::MyString(const char\* input) {  int len = strlen(input);  this->content = new char[len + 1];   strcpy(this->content, input);  this->content[len] = '\0'; } MyString::MyString(const MyString& from) {  int len = strlen(from.content);  this->content = new char[len + 1];  strcpy(this->content, from.content);  this->content[len] = '\0'; }  MyString& MyString::operator=(const MyString& from) {  if(this != &from)  {  delete[] content;   int len = strlen(from.content);  this->content = new char[len + 1];  strcpy(this->content, from.content);  this->content[len] = '\0';  }  return \*this; }  MyString::~MyString() {  delete[] this->content; }  void MyString::append(char to\_append) {  int len = strlen(this->content);  char\* new\_content = new char[len + 2];  strcpy(new\_content, this->content);  new\_content[len] = to\_append;  new\_content[len + 1] = '\0';  delete[] this->content;  this->content = new\_content; }  void MyString::print() {  std::cout << this->content << std::endl; } |
| --- |
|  |

## **Задача 2 (2 точки)**

Рефакторирайте *IntCounter* от първото домашно, да работи с темплейти, по дадения код:

*IntCounter.hpp*

| class IntCounter { private:  int\* ptr;  int\* counter;   void add\_reference();  void remove\_reference();  void freeMemory();   void redirectPointers(int\* newPtr, int\* newCounter);  void redirectPointers(const IntCounter& rhs);  public:  IntCounter();    IntCounter(int\* newPtr);  IntCounter(const IntCounter& rhs);  IntCounter& operator=(const IntCounter& rhs);  ~IntCounter();   int get\_count() const;  int get\_value() const; }; |
| --- |

## *IntCounter.cpp*

| IntCounter::IntCounter(): ptr(nullptr), counter(new int(0)) {} void IntCounter::add\_reference() {  \*(this->counter) += 1; }  void IntCounter::remove\_reference() {  \*(this->counter) -= 1; }  void IntCounter::freeMemory() {  delete this->ptr;  delete this->counter; }  void IntCounter::redirectPointers(int\* newPtr, int\* newCounter) {  this->ptr = newPtr;  this->counter = newCounter; }  void IntCounter::redirectPointers(const IntCounter& rhs) {  this->redirectPointers(rhs.ptr, rhs.counter); }  IntCounter::IntCounter(int\* newPtr) {  this->redirectPointers(newPtr, new int(1));  // this->ptr = ptr;  // this->counter = new int(1); }  IntCounter::IntCounter(const IntCounter& rhs) {  this->redirectPointers(rhs);  this->add\_reference(); }  IntCounter& IntCounter::operator=(const IntCounter& rhs) {  // Two objects are the same if they have the same pointer int\* ptr,  // which is different from the usual check in operator=  if (this->ptr != rhs.ptr) {  // Remove reference from the old pointer and check if deletion is due  this->remove\_reference();  if (this->get\_count() == 0) {  this->freeMemory();  }   // Point the variables to the new pointer and add a reference  this->redirectPointers(rhs);  this->add\_reference();  }  return \*this; }  IntCounter::~IntCounter() {  this->remove\_reference();  if (this->get\_count() == 0) {  this->freeMemory();  } }  int IntCounter::get\_count() const {  return \*this->counter; }  int IntCounter::get\_value() const {  return \*this->ptr; } |
| --- |

## **Задача 3 (4,5 точки)**

Реализирайте примитивна версия на стандарта *JSON*. В нашият *JSON* ще имаме два типа обекти - *JSONObject*, който е двойка от ключ (низ) и някаква стойност. Другият тип обект е *JSONArray* - той ще държи в себе си динамичен масив от обекти на класа *JSONObject*. Освен това, *JSONArray* трябва да може да добавя нови обекти по ключ и стойност, да премахва елементи по ключ, както и да връща стойност, отговаряща на даден ключ.

**Приема се, че няма да се въвеждат дублиращи се ключове.**

## **Задача 4 (1,5 точки)**

За класът *JSONArray* напишете функция: за записване на *JSONArray* във файл. **Функцията да пише в края на файла, който е отворен.**

*Сами изберете формата на файловете*

Ограничения и изисквания

* Предаване на домашното в указания срок от всеки студент като .zip архив със следното име: **(номер\_на\_домашно)**\_SI\_**(курс)**\_**(група)**\_**(факултетен\_номер)**, където:
  + **(номер\_на\_домашно)** е цяло число, отговарящо на номера на домашното за което е отнася решението (например 2);
  + **(курс)** е цяло число, отговарящо на курс (например 1);
  + **(група)** е цяло число, отговарящо на групата Ви (например 1);
  + **(факултетен\_номер)** e цяло число, отговарящо на факултетния Ви номер (например 63666);
* Архивът да съдържа само изходен код (.cpp и .h/.hpp файлове) с решение отговарящо на условията на задачите, като файловете изходен код за всяка задача трябва да са разположени в папка с име (номер\_на\_задача), където (номер\_на\_задача) е номера на задачата към която се отнася решението;
* Не е разрешено да ползвате класове от библиотеката STLкато std::string, std::vector, std::stack и др.
* Качване на архива на посоченото място в Moodle;

Пример за .zip архив за домашно: 2\_SI\_1\_1\_63666.zip