СУ „Св. Климент Охридски“, ФМИ

Специалност „Софтуерно Инженерство“

Обектно-ориентирано програмиране, 2019-2020 г.

Задача за домашно № 3

Спазвайте практиките за обектно-ориентирано програмиране, коментирани на упражнения и лекции. Позволено е използването на STL

Задача 1 (5 точки)

Довършете JSON задачата. Рефакторирайте кода на класовете `JSONArray` и `JSONObject` така че, `JSONArray` да може да им деца от тип `JSONArray` или `JSONObject`

JSONObject

```
#pragma once
#include <cstring>
#include <iostream>
template <class T>
class JSONObject{
private:
   char* key;
   T value;
public:
   JSONObject();
   JSONObject(const JSONObject& from);
   JSONObject<T>& operator=(const JSONObject& from);
   ~JSONObject();

   void set_key(const char* key);
   void set_value(T value);

   const char* get_key() const;
   T get_value() const;
};
```

```
template <class T>
JSONObject<T>::JSONObject()
{
   this->key = new char[1];
   this->key[0] = '\0';
   this->value =  T(); //Това принципно е лоша практика, но до C++ 20,
няма какво да се прави по въпроса
}

template <class T>
JSONObject<T>::JSONObject(const JSONObject& from)
{
   this->key = new char[strlen(from.key) + 1];
   strcpy(this->key, from.key);
   this->value = from.value;
}

template <class T>
JSONObject<T>& JSONObject<T>::operator=(const JSONObject& from)
{
   if(this != &from)
   {
       this->set_key(from.key);
       this->set_value(from.value);
   }
   return *this;
}

template <class T>
JSONObject<T>::~JSONObject()
{
   delete[] this->key;
}

template <class T>
void JSONObject<T>::set_key(const char* key)
{
   delete[] this->key;
   int len = strlen(key);
   this->key = new char[len + 1];
```

```
    strcpy(this->key, key);
    this->key[len] = '\0';
}

template <class T>
void JSONObject<T>::set_value(T value)
{
    this->value = value;
}

template <class T>
const char* JSONObject<T>::get_key() const
{
    return this->key;
}

template <class T>
T JSONObject<T>::get_value() const
{
    return this->value;
}
```

JSONArray

```
#include <cstring>

template <class T>
class JSONArray{
private:
    JSONObject<T>* array;
    int size;
    int capacity;

    void resize();
public:
    JSONArray();
    JSONArray(const JSONArray& from);

    JSONArray& operator= (const JSONArray& from);

    ~JSONArray();

    JSONObject<T> operator[] (const int& pos) const;
```

```
   void insert(const char* key, T value);

   T get_value(const char* key) const;

   int get_size() const;
};
template <class T>
void JSONArray<T>::resize()
{
      this->capacity *= 2;
      JSONObject<T>* new_array = new JSONObject<T>[this->capacity];

      for(int i = 0; i < this->size; i++)
      {
      new_array[i] = this->array[i];
      }

      delete[] this->array;
      this->array = new_array;
}
template <class T>
JSONArray<T>::JSONArray()
{
   this->array = new JSONObject<T>[1];
   this->size = 0;
   this->capacity = 1;
}

template <class T>
JSONArray<T>::JSONArray(const JSONArray& from)
{
   this->array = new JSONObject<T>[from.capacity];
   this->size = from.size;
   this->capacity = from.capacity;

   for(int i = 0; i < this->size; i++)
   {
      this->array[i] = from.array[i];
   }
}

template <class T>
JSONArray<T>& JSONArray<T>::operator= (const JSONArray& from)
{
   if(this != &from)
```

```
   {
      delete[] this->array;

      this->array = new JSONObject<T>[from.capacity];
      this->size = from.size;
      this->capacity = from.capacity;

      for(int i = 0; i < this->size; i++)
      {
            this->array[i] = from.array[i];
      }
   }
   return *this;
}

template <class T>
JSONArray<T>::~JSONArray<T>()
{
   delete[] this->array;
}

template <class T>
JSONObject<T> JSONArray<T>::operator[] (const int& pos) const
{
   return this->array[pos];
}

template <class T>
void JSONArray<T>::insert(const char* key, T value)
{
   if(this->size == this->capacity)
   {
      this->resize();
   }
   JSONObject<T>* temp = new JSONObject<T>();
   temp->set_key(key);
   temp->set_value(value);

   this->array[this->size] = *temp;
   this->size++;
}

template <class T>
T JSONArray<T>::get_value(const char* key) const
{
```

```
    for(int i = 0; i < this->size; i++)
    {
        if(strcmp(key, this->array[i].get_key()) == 0)
        {
                return this->array[i].get_value();
        }
    }
    std::cout << "key not found"; // Exception ?
    return T();
}

template <class T>
int JSONArray<T>::get_size() const
{
    return this->size;
}
```

Задача 2 (4 + 1 точки)

Имате задачата да напишете примитивна файлова система. В нашата файлове система, имаме следните типове обекти:

Файл:

- Име
- Дата (час, ден, месец и година) на създаване
- Размер в MB
- Разширение

Изображение - Разширява файл, като добавя следните характеристики:

- Име на устройството, с което е заснето изображението
- Резолюция (width/height)
- Разширение - `.jpg` или `.png`

Музикален файл - Разширява файл, като добавя следните характеристики:

- Име на изпълнителя
- Име на песента

- Година на песента
- Името на файла се генерира автоматично, по следният начин: `<име-на-изпълнител> - <име-на-песента>`
- Разширение - `.mp3` или `.flac`

Папка
- Име на папка
- Дата на създаване
- Папката може да съдържа други файлове или папки

Създайте клас **файлове система**, който да съдържа една главна папка, в която се съдържат всички други файлове и папки.
- Потребителят да може да създава файлове в дадена папка (може да подавате името на папката, в която да се създаде файла) - **Важно: това условие е за бонус точки. Без него може да изкарате максимума от 10 точки. Ако реализирате тази функционалност правилно, ще получите 1 бонус точка**
- Потребителят да може да търси файлове

Очаква се да напишете качествен код, с достатъчно добро разделяне на класовете.

Задача 3 (1 точки)

Моделирайте (напишете единствено класове, член-данни и сигнатурите на методите) на система за управление на студенти. Системата да поддържа курсове, оценки, студенти и преподаватели.

Курсовете имат:
- Име на курса
- Тип на курса
- Код на курса
- Преподавател
- Записани студенти

Студентите имат:
- Име
- Факултетен номер
- Записани курсове
- Оценки от изкарани курсове

Преподавателите имат:
- Име
- Титла (хоноруван, главен асистент, доктор, доцент, професор)
- Курсовете, които водят

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 3);
    - **(курс)** е цяло число, отговарящо на курс (например 1);
    - **(група)** е цяло число, отговарящо на групата Ви (например 1);
    - **(факултетен_номер)** е 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;
- Разрешено да ползвате класове от библиотеката STL като std::string, std::vector, std::stack и др.
- Качване на архива на посоченото място в Moodle;

Пример за .zip архив за домашно: 3_SI_1_1_63666.zip