Обектно-ориентирано програмиране
Домашна работа №1

Пояснение:

- Реализирайте задачите спазвайки добрите ООП практики(валидация на данните, подходяща капсулация и тн.)
- Решение, в които не са спазени ООП принципите ще бъдат оценени с 0 точки.
- Предадените от вас решения трябва да могат да се компилират успешно на Visual C++ или GCC.
- **Не е разрешено** да ползвате библиотеки от STL и STL функции.

Изисквания за предаване:

- Всички задачи ще бъдат проверени автоматично за преписване. Файловете с голямо съвпадение ще бъдат проверени ръчно и при установено плагиатство ще бъдат анулирани.
- Предаване на домашното в указания срок от всеки студент като .zip архив със следното име:

(номер_на_домашно)_SI_(курс)_(група)_(факултетен_номер)

  * (номер_на_домашно) е цяло число, отговарящо на номера на домашното за което се отнася решението (например 1);
  * (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 12345);

Пример за .zip архив за домашно: 1_SI_1_1_12345.zip

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).

Качване на архива на посоченото място в Moodle;

Обектно-ориентирано програмиране
Домашна работа №1

Задача 1: Hex viewer

Да се реализира програма за изглед и модификация на двоични файлове (**hex viewer**).
При стартиране на програмата трябва да се въведе път до двоичен файл.
След като файлът се е заредил в паметта, трябва да поддържате следните операции:

- Преглед на файла (**view**)
  - отпечатва байтовете на файла (в шестнайсетична бройна система).
  - След това да се отпечатат интерпретацията на байтовете като символи. Ако байтът отговаря на малка/голяма латинска буква, то да се отпечата символа.В противен случай да се отпечатва точка.
- Промяна на байт по индекс (**change <index> <value>**).
- Премахване на последния байт (**remove**)
- Добавяне на байт в края (**add <value>).**
- Запазване на промените в същия файл (**save**)
- Запазване на промените в друг файл (**save as <file name>**)

Обектно-ориентирано програмиране
Домашна работа №1

Примери:

Нека имаме двоичен файл, *myData.dat*, който се е получил след изпълнението на следния c++ код:

```
int x = 25409
ofstream file(“myData.dat”, ios::binary);
file.write( (const char*)&x, sizeof(x));
```

Пример за работа с програмата:

```
Enter a file path:
> myData.dat.
File loaded successfully! Size: 4 bytes
> view
41 63 00 00
A  c  .  .
>change 1 65
Operation successfully executed!
> view
41 65 00 00
A  e  .  .
> remove
> view
41 65 00
A  e  .
>save
File successfully saved
```

Софийски университет "Св. Климент Охридски"

Факултет по математика и информатика

Обектно-ориентирано програмиране
Домашна работа №1

Задача 2: Xml FMI

Напишете програма за работа с FMI_XML файлове. Тези файлове ще съдържат записи за студенти.

Информацията за всеки запис ще се пази между отварящ таг <Student> и затварящ таг <\Student>. Информацията за всяко поле ще се пази в съответните тагове.

- Име (низ до 25 символа) <name> <\name>
- Факултетен номер (цяло число) <fn> <\fn>
- Възраст (цяло число в интервала [15, 65]) <age> <\age>
- Пол (male/female) <gender> <\gender>
- Поща (низ до 25 симовола, който съдържа символа ‘@’) <email><\email>
- Среден успех (число с плаваща забетая в интервала [2, 6]) <grade> <\grade>

Всяко от тези полета е задължително!

Факултетният номер е уникален за всеки студент.

Полетата може да са разбъркани в произволен ред във файла.

Във файлът можете да имате неограничен брой интервали и табулации между таговете (но не и в съдържанието на самия таг).

Обектно-ориентирано програмиране
Домашна работа №1

Напишете програма, която зарежда такъв файл в паметта, и позволява следните операции:

- промяна на произволно поле на студент по подаден факултетен номер, име на поле и нова **валидна** стойност.
- сортиране на студентите по произволно поле. Ако полето е ‘пол’, можете да приемете, че всяко момиче трябва да е преди всяко момче.
- Отпечатване на информацията за студентите
- Запазване на информацията във файла, но във **форматиран** вид. Т.е вътрешните тагове да са табулация навътре и всички полета да имат еднакъв ред при всеки запис на студент.

Примери:

Съдържание на students.xml-fmi:

```
<student>
<grade>5.00<\grade>
<name>Petur Popov<\name>
<fn>2<\fn>
<age>18<\age>
<gender>Male<\gender><email>pepi@abv.bg<\email>
<\student>

<student>
<age>33<\age> <name>Ivan Petrov<\name>
<fn>1<\fn> <gender>Male<\gender>
<email>ivan@abv.bg<\email>
<grade>3.50<\grade>
 <\student>
```

Примери:

Обектно-ориентирано програмиране
Домашна работа №1

```
Enter a file path:
> students.xml-fmi.
File loaded successfully!
> edit 2 grade 3.00
Operation successfully executed!
> edit 1 email ivan
Error! Data not in correct format!
>sort age
Operation successfully executed!
>save
```

Примери:

Съдържание на students.xml-fmi след работата с файла :

```
<student>
        <name>Petur Popov<\name>
        <fn>2<\fn>
        <age>18<\age>
        <gender>Male<\gender>
        <email>pepi@abv.bg<\email>
        <grade>3.00<\grade>
<\student>

<student>
        <name>Ivan Petrov<\name>
        <fn>1<\fn>
        <age>33<\age>
        <gender>Male<\gender>
        <email>ivan@abv.bg<\email>
        <grade>3.50<\grade>
 <\student>
```