Обектно-ориентирано програмиране
Домашна работа №1

Пояснение:

- Реализирайте задачите, спазвайки добрите ООП практики (валидация на данните, подходяща капсулация и т.н.)
- Решения, 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 **административната Ви група** (например 1);
  - (факултетен_номер) е низ, отговарящ на факултетния Ви номер (например 12345 или 1MI01234);

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: **HTML tables parser**

Напишете програма за работа с HTML таблици. Трябва да работите със следните HTML тагове:

- <table> - Дефинира таблицата
- <th> - Дефинира заглавно поле в таблицата (не е задължително най-отгоре)
- <tr> - Дефинира ред в таблицата
- <td> - Дефинира клетка в таблицата

Не се изисква работа с атрибутите на тези тагове.
Трябва да поддържате работа с алтернативна кодировка на символите *(character entity reference)* - например &#97 съответства на символ ‘a’.

Трябва да поддържате следните функционалности:

(Номерата на редовете и колоните започват от 1.)

- Добавяне на ред в таблицата
  - add <rowNumber> <value1> <value2>...
- Премахване на ред от таблицата
  - remove <rowNumber>
- Промяна на клетка по номер на ред и колона
  - edit <rowNumber> <colNumber> <newValue>
- Визуализация на таблицата на конзолата
  - print

Забележки:

- Интерфейсът на командите е примерен, не е нужно да се придържате към него. Достатъчно е да е удобно за ползване.
- Клетките могат да съдържат всякакви възможни символи.

Програмата трябва да работи с текстови файлове, съдържащи HTML кода на конкретна таблица. Не е гарантирано, че всеки таг ще е на нов ред - програмата трябва да може да се

справи и с този случай. След успешно зареждане на таблица от файл програмата трябва да може да я манипулира чрез горепосочените функционалности.

От конзолата ще прочетете път към текстов файл с код на таблица, както и серия от команди за манипулация. При получаване на команда за принтиране, изведете таблицата в следния формат:

```
|*header1*|*header2*| ... |*headerN*|
| data11  | data12  | ... | data1N  |
| data21  | data22  | ... | data2N  |
```

Ако някой от редовете има по-малък брой клетки от останалите, изведете празни клетки, за да бъде подравнена коректно таблицата в конзолата.

Примерно съдържание на файл и очаквана резултатна таблица:

```
<table>
  <tr>
    <th>N&#97me</th>
    <th>Age</th>
    <th>Fn</th>
  </tr>
  <tr>
    <td>Petur Ivanov</td>
    <td>34</td>
    <td>12345</td>
  </tr>
  <tr>
    <td>Maria Petrova</td>
    <td>12</td>
  </tr>
  <tr>
    <td>Stefan Stefanov</td>
    <td></td>
    <th>1234</th>
```

```
  <td>Unknown field</td>
 </tr>
</table>
```

Name	Age	Fn	
Petur Ivanov	34	12345	
Maria Petrova	12		
Stefan Stefanov		**1234**	Unknown field

```
|*Name            *|*Age*|*Fn   *|                |
| Petur Ivanov     | 34  | 12345 |                |
| Maria Petrova    | 12  |       |                |
| Stefan Stefanov  |     |*1234 *| Unknown field |
```

Може да тествате визуализацията на HTML таблиците тук: W3Schools HTML Tables

Ограничения: Една клетка може да съдържа най-много 50 символа след обръщане на character entity reference-ите в обикновени ASCII символи. Един ред може да съдържа до 30 клетки, а една таблица може да съдържа до 300 реда.

Задача 2: **Playlist**

Да се имплементира система, която управлява плейлист с песни.
Всяка песен има:

- Име (до 64 символа)
- Продължителност
- Жанр - може да бъде прост жанр: рок, поп, хип-хоп, електронна музика, джаз или комбинация между 2 или повече прости жанра (напр. рок и поп)
- Съдържание - последователност от байтове (най-много 256), която се прочита от двоични файлове

Обектно-ориентирано програмиране
Домашна работа №1

Жанрът трябва да се пази в структура/член-данна, която заема **най-много 1 байт.**

Съдържанието на песента трябва да може да се модифицира по 1 от следните начини:

- Всеки k-ти бит отзад напред да се преобразува в 1-ца (за създаване на допълнителен ритъм)

Пример:
0100 0110, 3 -> 01**1**0 0**1**10

- Да се миксира с друга песен - 1-ци остават всички битове, които са 1-ци в **само една** от двете песни (ако някой бит е 1-ца и в двете песни, то ще бъде 0-ла в резултата). Ако една песен се миксира с друга с различна дължина, се миксират толкова байтове, колкото е дължината на по-кратката.

Пример:
Песен 1: 0110 0001
Песен 2: 0101 0101
-> Песен 1: 0011 0100

Песен 1: 0101 1010 0101 1010
Песен 2: 0111 0111
-> Песен 1: 0010 1101 0101 1010

Песен 1: 0111 0111
Песен 2: 0101 1010 0101 1010
-> Песен 1: 0010 1101
Уточнение: При миксиране на дадена изходна песен с друга, миксът презаписва съдържанието на изходната песен, като дължината на съдържанието не се променя.

Плейлистът може да съдържа най-много 30 песни. Трябва да се поддържат следните функционалности:

- Добавяне на песен в плейлиста с определено име, продължителност, жанр и име на файл, от който се чете съдържанието. Жанрът се подава като стринг, където всяка буква отговаря на даден прост жанр (напр. “гр” означава рок и поп)
- Принтиране на песните в плейлиста - име, продължителност във формат часове:минути:секунди и жанр
- Търсене на песен по име (всяка песен гарантирано е с уникално име)
- Търсене на песен по жанр - връща всички песни, които съдържат определен жанр

Обектно-ориентирано програмиране
Домашна работа №1

- Сортиране по продължителност на песента
- Сортиране по име на песента
- Модифициране на песен по някой от посочените начини (по подадено име)
- Запазване на дадена песен в двоичен файл (по подадено име на песента и име на двоичен файл

Примерен интерфейс:

Съдържание на **song1.dat:**
“V“
// 0101 0110

Съдържание на **song2.dat:**
“Ua“
// 0101 0101 0110 0001

Playlist p;
p.add(“Song 2”, 0, 1, 55, “rp”, “song2.dat“);
p.add(“Song 1”, 0, 1, 5, “p”, “song1.dat“);

p.print();
// Song 2, 00:01:55, Pop&Rock
// Song 1, 00:01:05, Pop

p.sortByName();
p.print();
// Song 1, 00:01:05, Pop
// Song 2, 00:01:55, Pop&Rock

p.find(“Song 1”);
// Song 1, 00:01:05, Pop

p.findGenre(‘p’);
// Song 1, 00:01:05, Pop
// Song 2, 00:01:55, Pop&Rock
p.findGenre(‘r’);
// Song 2, 00:01:55, Pop&Rock

p.mix(“Song 1”, “Song 2”);
p.save(“Song 1”, “song3.dat”);
// съдържание на song3.dat: “S” //0000 0011