Practicum 1 - deel 2: configuratiebestand

In dit practicum schrijf je programmacode om configuratiebestanden in te lezen. Je maakt hiervoor een "ConfigReader" klasse, waarvan de interface er als volgt uitziet:

```
class ConfigReader
{
public:
     ConfigReader();
     ~ConfigReader();

     bool read(const std::string &fileName);
     bool getKeyValue(const std::string &key, std::string &value);
     void printAll() const;
     void clear();

     std::string getErrorDescription() const;
};
```

De functies die `bool` als returnwaarde hebben, leveren `true` als de functie gelukt is, en `false` in het andere geval. Als er iets misgaat wordt er bovendien intern een foutbeschrijving opgeslagen die je kan opvragen met behulp van de `getErrorDescription` member functie.

De `read` functie staat in voor het inlezen van een bestand met key-value paren, dat er bijvoorbeeld als volgt uitziet:

```
rawfile=sample00.yuv
encfile=encoded.enc
width=352
height=288
rle=1
quantfile=matrix.txt
logfile=log.txt
gop=4
merange=16
```

Zorg er in de eerste plaats voor dat de file regel per regel kan ingelezen worden, zowel als er UNIX ('\n') als DOS ('\r\n') tekens gebruikt worden om de regels van elkaar te scheiden. De 'key' waarde is dan alles wat er voor het eerste '=' teken staat op die regel, inclusief eventuele spaties. De 'value' is alles wat er achter komt, maar zonder de '\n' of '\r\n' (zou dus in principe zelfs '=' tekens kunnen bevatten als er meer dan één staat in de regel). Een regel die volledig leeg is mag niet als fout beschouwd worden, maar een regel die bijvoorbeeld enkel een spatie bevat wel. Een lege key-waarde moet ook voor een fout zorgen, alsook het voorkomen van twee dezelfde key-waarden. Een lege value is wel toegestaan. De 'value' waarden moeten gewoon als strings opgeslagen worden, er moet geen interpretatie van gebeuren in deze klasse.

De `getKeyValue` functie moet controleren of de opgegeven key bestaat, en indien dit het geval is, `value` correct invullen. De `printAll` functie toont alle key-value paren op het scherm, in hetzelfde formaat als het configuratiebestand, dus: `key=value`. Lege regels die in het oorspronkelijk bestand aanwezig waren moeten uiteraard niet opnieuw uitgeschreven worden, en de volgorde van key-value paren mag ook verschillen. Het is m.a.w. niet de bedoeling dat deze functie het ingelezen bestand byte voor byte toont, wel dat de opgeslagen informatie getoond wordt in een soortgelijk formaat. De `clear` functie wist alle opgeslagen key-value paren.

De volgende code illustreert hoe de klasse gebruikt kan worden:

```
#include "configreader.h"
#include <iostream>

int main(int argc, char *argv[])
{
     if (argc != 2)
     {
          std::cerr << "Usage: configreader filename" << std::endl;
          return -1;
     }

     ConfigReader cfg;

     if (!cfg.read(argv[1]))
     {
          std::cerr << cfg.getErrorDescription() << std::endl;
          return -1;
     }

     cfg.printAll();
     std::cout << std::endl;

     std::string value;

     if (!cfg.getKeyValue("encfile", value))
          value = "ERROR: " + cfg.getErrorDescription();

     std::cout << "Encoded file: " << value << std::endl;

     if (!cfg.getKeyValue("decfile", value))
          value = "ERROR: " + cfg.getErrorDescription();

     std::cout << "Decoded file: " << value << std::endl;

     return 0;
}
```