Wprowadzenie

Załóżmy, że istnieje pewien pojazd wyposażony w komputer pokładowy, rejestrujący przebieg jazdy. Tuż po uruchomieniu pojazdu tworzony jest plik tekstowy, w którym zapisywane są znaki, określające zdarzenia zachodzące podczas jazdy. Po zakończeniu jazdy, plik jest zamykany. Nas interesują zdarzenia: przyśpieszanie (oznaczane znakiem a lub A), zwalnianie (oznaczane znakiem b lub B), skręt w lewo (oznaczany znakiem l lub L), skręt w prawo (oznaczany znakiem r lub R). W pliku mogą wystąpić inne znaki, oznaczające zdarzenia, które nas w tym momencie nie interesują.

Przykładowa zawartość pliku:

```
aaxabbvllllyrrsedieeofefefefaaabbbbllrrrr
az
bz
lz
rz
```

Nas interesują wskazane wyżej znaki:

```
aaabbllllrraaabbbbllrrrrablr
```

Na ich podstawie możemy stwierdzić ile razy przyśpieszano, zwalniano, skręcano w lewo i w prawo. W przedstawionym przykładzie było to dla każdego ze zdarzeń siedem razy.

Zadanie

Naszym zadaniem jest zdefiniowanie klasy, która na podstawie zawartości pliku pozwoli wyznaczenie ile razy przyśpieszano, zwalniano, skręcano w lewo i w prawo. Klasa ma posiadać następujące publiczne funkcje składowe:

```
class RideReport
{
   public:
 RideReport(const char *fileName);
 ~RideReport();
 bool readData();
 bool processData();
 int getSpeedUpCount();
 int getSlowDownCount();
 int getLeftTurnCount();
 int getRightTurnCount();
 bool saveCleanedData(const char *fileName);
 bool saveAsText(const char *fileName);
 bool saveAsXML(const char *fileName);
  private:
 // Składowe prywatne wg potrzeb i własnego uznania programisty
};
```

Poniższy przykład pokazuje jak może być wykorzystana klasa:

```
...
int main()
{
 RideReport report("dane.txt");
 if(report.readData())
 {
  if(report.processData())
  {
   cout << "\nPrzyspieszenia: " << report.getSpeedUpCount();
   cout << "\nHamowania: " << report.getSlowDownCount();
   cout << "\nW lewo: "  << report.getLeftTurnCount();
   cout << "\nW prawo: " << report.getRightTurnCount();
   if(report.saveCleanedData("dane_czyste.txt"))
    cout << "\nOczyszczone dane zapisane";
   if(report.saveAsText("raport.txt"))
    cout << "\nRaport tekstowy zapisany";
   if(report.saveAsXML("raport.xml"))
    cout << "\nPlik XML zapisany";
  }
 }
 return
EXIT_SUCCESS;
}
```

Wymagane zachowanie funkcji składowych

`RideReport(const char *fileName)`

Konstruktor, inicjalizacja klasy, parametr *fileName* to nazwa pliku z danymi. Konstruktor nie otwiera pliku, zapamiętuje nazwę. Nazwa pliku ma zostać zapamiętana wewnętrznie przez obiekt klasy *RideReport* w prywatnej, dynamicznie alokowanej tablicy znaków, o rozmiarze ustalanym wg rozmiaru parametru *fileName*.

`~RideReport()`

Destruktor, wykonuje czynności deinicjalizujące, zależne od szczegółów implementacyjnych klasy, np. zwalnianie zaalokowanej pamięci.

Informacje o destruktorach:

- https://docs.microsoft.com/pl-pl/cpp/cpp/destructors-cpp?view=vs-2019
- https://www.geeksforgeeks.org/destructors-c/
- https://beginnersbook.com/2017/08/cpp-destructors/

`bool readData()`

Funkcja otwiera plik o nazwie przekazanej przez parametr konstruktora (zapamiętana nazwa), ładuje zawartość pliku do pamięci, wykorzystując dynamicznie alokowaną tablicę znaków. Rozmiar tablicy znaków ma odpowiadać wielkości pliku.

Alokujemy tylko tyle pamięci, aby zmieściła się zawartość pliku. Rezultatem funkcji ma być wartość *true* jeżeli operacja otwarcia pliku i załadowania jego zawartości zakończyła się bezbłędnie, *false* w przeciwnym przypadku. Plik z danymi nie jest otwierany w żadnej innej funkcji klasy *RideReport*.

```
bool processData()
```

Zadaniem funkcji jest przetworzenie danych odczytanych z pliku przez funkcję *readData* (zapamiętanych przez nią w tablicy). Funkcja (na podstawie zbuforowanych w tablicy danych) wyznacza ile razy pojazd przyśpieszył, zwolnił, skręcił w lewo, w prawo.

```
int getSpeedUpCount()
```

Rezultatem funkcji ma być liczba zarejestrowanych przyśpieszeń lub wartość -1 jeżeli nie udało się przetworzyć pliku z danymi wejściowymi.

```
int
```

```
getSlowDownCount()
```

Rezultatem funkcji ma być liczba zarejestrowanych zwolnień lub wartość -1 jeżeli nie udało się przetworzyć pliku z danymi wejściowymi.

```
int
```

```
getLeftTurnCount()
```

Rezultatem funkcji ma być liczba zarejestrowanych skrętów w lewo lub wartość -1 jeżeli nie udało się przetworzyć pliku z danymi wejściowymi.

```
int getRightTurnCount()
```

Rezultatem funkcji ma być liczba zarejestrowanych skrętów w prawo lub wartość -1 jeżeli nie udało się przetworzyć pliku z danymi wejściowymi.

```
bool saveCleanedData(const char *fileName)
```

Funkcja zapisuje do pliku o nazwie *fileName* dane oczyszczone z nieinteresujących nas znaków. Zatem dla podanej wyżej, przykładowej zawartości pliku wejściowego, funkcja powinna zapisać do wskazanego pliku następującą zawartość:

```
aaabbllllrraaabbbbbllrrrrablr
```

```
bool saveAsText(const char *fileName)
```

Funkcja zapisuje do pliku o nazwie *fileName* raport w postaci tekstu w następującej formie:

```
Raport dla pliku:
<nazwa pliku z danymi>
Data utworzenia:
<data utworzenia raportu w formacie rrrr-mm-dd>
Czas utworzenia:
<czas utworzenia pliku w formacie gg:mm:ss>
Przyśpieszenia:
<liczba zarejestrowanych przyśpieszeń>
Hamowania: <liczba zarejestrowanych zwolnień>
Lewo: <liczba zarejestrowanych skrętów w lewo>
Prawo: <liczba zarejestrowanych skrętów w prawo>
```

Dla podanej wyżej, przykładowej zawartości pliku wejściowego, funkcja powinna zapisać do wskazanego pliku następującą zawartość:

```
Raport dla pliku:
dane.txt
Data utworzenia:
2020-03-16
Czas utworzenia:
18:26:38
Przyśpieszenia: 7
Hamowania: 7
Lewo: 7
Prawo: 7
```

Rezultatem funkcji jest wartość *true* jeżeli udało się zapisać raport zawierający informacje odczytane przez funkcję *readData* i wyznaczone przez funkcję *processData*. W przypadku, gdy utworzenie raportu nie było możliwe, rezultatem funkcji jest wartość *false*.

```
bool saveAsXML(const char *fileName)
```

Funkcja zapisuje do pliku o nazwie *fileName* raport w postaci tekstu w formacie XML:

```
<?xml version="1.0" encoding="utf-8"?>
<report file="nazwa pliku z danymi" date="data utworzenia
raportu w formacie rrrr-mm-dd" time="czas utworzenia pliku w formacie
gg:mm:ss">
<speed-up-count>liczba zarejestrowanych przyśpieszeń</speed-up-count>
<slow-down-count>liczba zarejestrowanych zwolnień</slow-down-count>
<left-turn-count>liczba zarejestrowanych skrętów w lewo<left-turn-count>
<right-turn-count>liczba zarejestrowanych skrętów w prawo<right-turn-
count>
</report>
```

Dla podanej wyżej, przykładowej zawartości pliku wejściowego, funkcja powinna zapisać do wskazanego pliku następującą zawartość:

```
<?xml version="1.0" encoding="utf-8"?>
<report file="dane.txt" date="2020-03-16" time="18:26:38">
<speed-up-count>7</speed-up-count>
<slow-down-count>7</slow-down-count>
<left-turn-count>7<left-turn-count>
<right-turn-count>7<right-turn-count>
</report>
```

Rezultatem funkcji jest wartość true jeżeli udało się zapisać raport zawierający informacje odczytane przez funkcję *readData* i wyznaczone przez funkcję *processData*. W przypadku, gdy utworzenie raportu nie było możliwe, rezultatem funkcji jest wartość *false*.

Data i czas tworzenia raportów

Wskazówka w kwestii wyznaczania daty i czasu:

https://www.tutorialspoint.com/cplusplus/cpp_date_time.htm

Operacje plikowe

Do realizacji operacji plikowych wystarczą standardowe funkcji znane z języka C: *fopen*, *fclose*, *fread*, *fwrite*. Te funkcje "przerabialiśmy" we wcześniejszych zadaniach. Jeżeli ktoś chce użyć obiektowych operacji na plikach (istream, ostream) proszę bardzo. Wskazówki:

- https://www.w3schools.com/cpp/cpp_files.asp
- https://www.cprogramming.com/tutorial/c++-iostreams.html
- https://www.tutorialspoint.com/cplusplus/cpp_files_streams.htm

Uwagi

Proszę się zastanowić, czy klasa *nie powinna posiadać jeszcze jakiegoś innego konstruktora*, jeżeli ktoś uzna, że tak, to proszę dodać taki konstruktor i w komentarzu uzasadnić dlaczego powinien być. Czy może w klasie *brakuje jeszcze jakiejś potrzebnej funkcji publicznej*? Jeżeli tak, to proszę dopisać funkcję oraz w komentarzu napisać uzasadnienie dlaczego jest konieczna.

Funkcje klasy RideReport nie wypisują żadnych komunikatów, mają robić dokładnie to, co zostało wskazane w przedstawionej wcześnie specyfikacji, oraz czynności niezbędne do osiągnięcia tego celu.

Testowanie

Proszę opracować testy i przetestować działanie klasy, minimalny zestaw testów obejmować powinien przypadku:

1. przekazanie konstruktorowi wskaźnika pustego,
2. przekazanie konstruktorowi nazwy nieistniejącego pliku,
3. przekazanie konstruktorowi nazwy pustego pliku,
4. przekazanie konstruktorowi nazwy pliku nie zwierającego znaków żadnego ze znaków z zestawu aAbBlLrR.

Można przeprowadzić eksperymenty z bardzo dużymi plikami, ciekawe wydaje się ustalenia dla jak dużych plików będzie brakowało pamięci, aby zaalokować tablicę buforującą dane z pliku.

Rozwiązanie

Jako rozwiązanie proszę przesyłać plik *zip* zawierający kod źródłowy programu oraz pliki z danymi na jakich przeprowadzono testy. Kod źródłowy powinien zawierać kod klasy oraz kod realizujący testy. Plik z rozwiązaniem:

z9_imie_nazwisko.zip

Proszę nie używać polskich liter w nazwach plików.