Mittuniversitetet
MID SWEDEN UNIVERSITY

Lab 4 – Exceptions, mallar och strömmar

Objektorienterad programmering i C++

Syfte: Tillämpning av undantagshantering i kombination med mallar och IO-strömmar.

Lab 4 – Exceptions, mallar och strömmar

Bakgrund
Du ska utgå från följande scenario: Mätvärden från en extern givare lagras som numeriska värden i en fil på disken. Mätningarna och överföringen är mycket störkänsliga vilket leder till att de lagrade mätvärdena innehåller två typer av fel:

1. *Givarfel* som medför att värdena ligger utanför ett specificerat intervall.

2. *Överföringsfel* som medför att andra tecken än siffror skrivs till filen.

Uppgift
Din uppgiften är att skriva ett program som fitrerar mätvädena i två steg. Först filtreras bort tal som inte kan läsas p.g.a. att de innehåller felaktiga tecken, och därefter filtreras bort de tal som ligger utanför det accepterade intervallet.

De värden som ska behandlas finns i filen `values.dat`. Talen är skrivna i textformat och åtskiljs av newline-tecken. Värdena ska i programmet konverteras till typen double. I filtreringsprocessen ska undantagshantering användas.

Läsningen av värden från `values.dat` och filrering av felaktiga tecken ska göras med en instans av klassen DataFileReader:

```
template<typename T>
class DataFileReader {

public:
  DataFileReader(string aDataFileName, string aErrorFileName);
  /* pre: A file named aDataFile contains values to read.
  */

  ~DataFileReader();
  /* post: Files are closed */

  void openFiles();
  /* post:  An input stream from the file named aDataFile and
      an output stream to the file named aErrorFile are
      opened. If either of these operations fails a
      runtime_error exception is thrown. */

  bool readNextValue(T &aValue);
  /* pre:  openFiles has been successfully called.
      post: If a value has been successfully read, aValue
      holds that value and true is returned.
      Else, the read operation encountered an
      end-of-file and false is returned.  */


private:
      ... necessary members
};
```

Funktionen `readNextValue()` ska implementeras efter följande riktlinjer:

- Funktionen ska läsa **ett** värden från filen och försöka konvertera det till den typ som ges av den aktuella typparametern `T`. Funktionen ska inte hantera filtrering av värden utanför det specificerade intervallet.

- Feltillstånd i aktuella inputströmmen ska hanteras internt i funktionen med hjälp av `std::ios_base::failure` -exceptions från stream-objektet och try/catch-block. Om ett läst värde inte kan konverteras ska det i stället skrivas (som en sträng) till den fil som öppnats med parametern aErrorFile för att kunna anlyseras senare. Efter hanteringen av ett felaktigt värde ska en ny läsning med försök till konvertering göras. Detta upprepas tills funktionen kan returnera ett värde via utparametern aValue **eller** tills end-of-file har påträffats.

Filterering av lästa värden ska göras med en instans av klassen DataFilter

```
template<typename T>
class DataFilter {
public:
      DataFilter(DataFileReader<T> *aReader, T aMin, T aMax);
      /* pre:  aReader points to an instance of DataFileReader<T>
            for which openFiles() has been succesfully called.
      */

      bool getNextValue(T &aValue);
      /* pre:  an earlier call to getNextValue() has not returned
            false.
         post: true is returned if aValue holds a value read from
            aReader. If a value could not be read, false
            is returned. If a value is read but is not within
            the interval specified by aMin and aMax parameters
            to the constructor, a range_error exception is
            thrown.
      */
private:
      ... necessary members
};
```

DataFilter<T> ska alltså via konstruktorn konfigureras med en pekare till en DataFileReader<T> samt min/max-gränser för giltiga data. `getNextValue()` använder instansen av DataFileReader<T> för att hämta nästa värde från filen. Värden som ligger utanför det specificerade intervallet ska resultera i ett `std::range_error`-exception med en strängrepresentation av det ej accepterade värdet som parameter.

Ett testprogram ska använda klasserna beskrivna ovan för att läsa in värdena från Values.dat och redovisa ett resultat.

- Accepterade värden ska ligga inom intervallet [0.0 .. 10.0].
- Exceptions från DataFilter ska hanteras så att alla värden som ligger utanför intervallet skrivs till en fil med namnet RangeErrors.dat.
- De värden som inte kunde konverteras i DataFileReader ska skrivas till en fil med namnet ReadErrors.dat

- Programmet ska redovisa antalet, summan av och medelvärdet av de korrekt lästa värdena samt antalet värden som kunde läsas men som låg utanför det specificerade intervallet.

Krav på lösningen

- Angivna pre- och post conditions ska tillämpas.
- Medlemsfunktionen `readNextValue` ska internt utnyttja exceptions från input-strömmen.
- Lösningen ska vid provkörning ge korrekt resultat:
    - Lästa numeriska värden: 19773
    - Värden utanför intervallet: 201
    - Summa: 99702,5
    - Medelvärde: 5,04235