6 Excepties en afsluiter

Oefening 127

Op Minerva staat het bestand `127_tekstbestanden.zip` in de map `txt-bestanden`. Deze .zip bevat de bestanden die je nodig hebt voor deze opgave.

Schrijf een functie `regel_uit_bestand(bestand,volgnr)` die uit een gegeven bestand de regel tekst haalt die het gegeven volgnummer draagt. Deze functie gooit een exceptie van type `string` indien het bestand niet geopend kon worden, een exceptie van type `c-string` als het bestand niet start met het woord `VERHAAL`, en een exceptie van eigen makelij (type `bestand_niet_lang_genoeg`) als het bestand niet lang genoeg is. (Afspraak: de regel met het woord `VERHAAL` heeft regelnummer 0.)

Daarna pas je het hoofdprogramma dat hieronder gegeven is aan. Eerst worden alle gevraagde (en gevonden) regels onder elkaar uitgeschreven. Daaronder komt een lijst van alle bestanden die niet geopend konden worden, gevolgd door een lijst van alle bestanden die te kort waren. Tenslotte komen de eerste woorden uit de bestanden die niet begonnen met het woord `VERHAAL` - die vormen op zich weer een stukje tekst.

```
#include <iostream>
#include <fstream>
#include <vector>
#include <string>
using namespace std;

int main(){

    vector<string> bestandsnamen{"niks","een","twee","drie","vier",
                     "vijf","zes","zeven","acht","negen","tien","elf","twaalf"};
    vector<int> nrs{8,5,2,10,7,3,8,4,1,1,6,2,4};

    string bestanden_niet_gevonden = "";
    string bestanden_niet_lang_genoeg = "";
    string eerste_woorden = "";

    for(int i=0; i<bestandsnamen.size(); i++){
        cout << regel_uit_bestand(bestandsnamen[i]+".txt",nrs[i]) << endl;
    }

    cout<<endl<<endl<<"BESTANDEN NIET GEVONDEN:"<<endl;
    cout<<bestanden_niet_gevonden;

    cout<<endl<<"BESTANDEN NIET LANG GENOEG:"<<endl;
    cout<<bestanden_niet_lang_genoeg<<endl;

    cout<<endl<<"BESTANDEN ZONDER STARTWOORD 'VERHAAL':"<<endl;
    cout<<"dit waren de woorden die er wel als eerste stonden:"<<endl<<endl;

    cout<<eerste_woorden<<endl<<endl;

    return 0;
}
```

Het eindresultaat zal er dan als volgt uit zien:

```
... (tekstje)
```

print hier 1x "VERHAAL" uit, komt uit negen.txt

```
BESTANDEN NIET GEVONDEN:
vier.txt kon niet geopend worden
zes.txt kon niet geopend worden

BESTANDEN NIET LANG GENOEG:
drie.txt heeft geen 10 regels.

BESTANDEN ZONDER STARTWOORD 'VERHAAL':
dit waren de woorden die er wel als eerste stonden:

... (tekstje)
```

Oefening 128

De klassen `Schrijf` en `Doos` uit oefening 121 voldoen niet aan de **"rule of five"**.

Vul deze klassen aan zodat wel voldaan is aan deze regel.

Oefening 129

Gegeven een blokkendoos waarin je figuren kan steken zoals in een vector, met één verschil: de grootste van deze figuren (wat oppervlakte betreft) zal altijd apart bewaard worden. Vraag je de grootste figuur van de blokkendoos, dan verdwijnt die ook effectief uit de doos.

Om zeker te zijn elke figuur maar 1 keer worden toegekend gebruik je `unique pointer`.

Download de bestanden `figuren.h` en `figuren.txt` van Minerva.

Bekijk grondig de interface van de klasse `Figuur` en de inhoud van het bestand `figuren.txt`. Bekijk dan het onderstaand hoofdprogramma. Probeer eerst zonder de tips te voorspellen wat je moet implementeren. Pas daarna lees je verder wat onder de code verklapt wordt.

```
#include "figuren.h"
#include <memory>

class Blokkendoos : vector<unique_ptr<Figuur>>{
  private:
    unique_ptr<Figuur> max_opp;
    void schrijf(ostream&)const;

  public:
    Blokkendoos();
    Blokkendoos(const string & bestandsnaam);
    unique_ptr<Figuur> geef_figuur_met_grootste_oppervlakte();
    void push_back(unique_ptr<Figuur>& figuur);

    friend ostream& operator<<(ostream& out, const Blokkendoos& l){
        l.schrijf(out);
        return out;
    }
};

int main(){
    Blokkendoos blokkendoos("figuren.txt");
```

```
    cout<<endl<<"ALLE FIGUREN: ";
    cout<<blokkendoos<<endl;

    cout<<endl<<"DE 3 GROOTSTE, van groot naar klein: "<<endl;
    for(int i=0; i<3; i++){
        cout<<"figuur met grootste opp:    "<<*blokkendoos.geef_figuur_met_grootste_oppervlakte()<<endl;
    }

    cout<<endl<<"DE NIEUWE BLOKKENDOOS BEVAT ALLEEN NOG DE KLEINERE FIGUREN: ";
    cout<<blokkendoos<<endl;

    return 0;
}
```

Tips

1. Bij het toevoegen van een figuur `fig` aan de blokkendoos (`push_back`) maak je onderscheid tussen twee gevallen. Indien de grootste figuur nog niet gekend is (dan is de vector dus zeker nog leeg), initialiseer je deze grootste figuur met de figuur `fig`. In het andere geval voeg je de nieuwe figuur achteraan de vector toe. Indien die echter (qua oppervlakte) groter blijkt te zijn dan `max_opp`, dan wissel je beide.
2. De methode die de grootste figuur teruggeeft, moet er ook voor zorgen dat de blokkendoos een nieuwe grootste figuur aanduidt uit de voorraad die nog in de vector zit. Pas de grootte van de vector ook aan!