## 2 Unique pointers,λambdafuncties en containers

**Oefening 109**

```
//Schrijf de inhoud van 1 unieke pointer, indien niet de nullptr
void schrijf (const unique_ptr<string> &s){
    if(s==nullptr){
        cout<<"NULL";
    }
    else{
        cout<<*s;
    }
}

void schrijf(const unique_ptr<string> * s, int aantal){
    cout<<endl;
    for(int i=0; i<aantal-1; i++){
        schrijf(s[i]);
        cout<<" - ";
    }
    schrijf(s[aantal-1]);
}

// OPGELET!! WAT WE NIET WILLEN ZIEN STAAN IN DE PROCEDURE HIERONDER:
//    *s[i] = *s[i+1]   --> kopieert strings
// OF
//    *s[i] = move(*s[i+1]) --> swapt de strings; probeer eens uit:
//                als je 'Rein' weghaalt uit 'Rein Ada Eppo'
//                eindig je met 'Ada Eppo Rein'

void verwijder(unique_ptr<string> * s, int aantal, int volgnr){
    if(volgnr < aantal-1){
        for(int i = volgnr; i < aantal-1; i++){
            s[i] = move(s[i+1]);
        }
    }
    else if(volgnr == aantal-1){
        s[volgnr].reset();
    }
}

//Declaratie in het hoofdprogramma:

    unique_ptr<string> pnamen[]={
        make_unique<string>("Rein"),
        make_unique<string>("Ada"),
        make_unique<string>("Eppo"),
        make_unique<string>("Pascal"),
        make_unique<string>("Ilse")};
```

**Oefening 110**

```
#include <iostream>
#include <functional>
#include <iomanip> // voor setw
using namespace std;

void schrijf(const string & tekst, const int * v, int aantal){
    cout<<tekst;
    for(int i=0; i<aantal; i++){
        cout<<setw(4)<<v[i]<<" ";
    }
    cout<<endl;
}

void vul_array(const int * a, const int * b, int * c, int grootte,
   function<int(int,int)> func ){
    int i;
    for(i=0; i<grootte; i++){
        c[i] = func(a[i],b[i]);
    }
}

//aanroep in het hoofdprogramma:
    vul_array(a,b,c,GROOTTE,[](int x,int y){return x+y;});
    vul_array(a,b,c,GROOTTE,[](int x,int y){return x*y;});
    vul_array(a,b,c,GROOTTE,[](int x,int y){return x-y;});
```

---

**Oefening 111**

```
/* stap 1-3 */
template <typename T>
void schrijf(const vector<T> &v){
    for(int i=0; i< v.size();i++ ){
        cout<<v[i]<<" ";
    }
    cout<<endl;
}

// stap5: de schrijf-procedure voor vectoren wordt vervangen door
// de uitschrijf-operator voor vectoren:
template<typename T>
ostream& operator<<(ostream& out, const vector<T> & v){
    for(int i=0; i<v.size(); i++){
        out<<v[i]<<" ";
    }
    cout<<endl;
    return out;
}
```

**Oefening 112**

```
// Set uitschrijven? als je per se komma's tussen de elementen wil
// (en geen komma meer na het laatste element), kan je niet met een
// gewone while-lus werken. je kan niet 'rekenen' met iteratoren,
// dus " s.end() - 1 " heeft geen betekenis (al had je gehoopt dat dat
// de iterator zou zijn die op het laatste element staat te wijzen...)

// Wat wel zou kunnen: een iterator 'hulp' gelijkstellen aan s.end(),
// die iterator 'hulp' dan eentje naar voor schuiven, en in de
// while-lus voortdoen zolang de iterator die vooraan startte,
// niet gelijk is aan die iterator 'hulp'.

template<typename T>
ostream& operator<<(ostream& out, const set<T> & s){
     out<<"{ ";
     typename set<T>::const_iterator it = s.begin();
     for(int i=0; i<s.size()-1; i++){
          out<<*it<<" , ";
          it++;
     }
     out<<*it<<" }";
     return out;
}

template<typename T>
ostream& operator<<(ostream& out, stack<T> st){   // st MOET kopie
   zijn
     while(!st.empty()){
          out<<"  "<<st.top()<<endl;
          st.pop();
     }
     return out;
}

template<typename S, typename D>
ostream& operator<<(ostream& out, const map<S,D> & m){
     typename map<S,D>::const_iterator it = m.begin();
     while(it!=m.end()){
          out<<"  "<<it->first<<" --> "<<it->second<<endl;
          it++;
     }
     return out;
}
```

**Oefening 113**

```
#include "containers.h"

const int AANTAL = 5;

int main(){
//oefening 1
    stack<string> st;
    st.push("een");
    st.push("twee");
    st.push("drie");
    cout<<st; // als je geen kopie had gemaakt bij uitschrijven van
       de stack, zal dit
    cout<<st; // hier niet werken!!! (dan heb je de 2e maal een lege
       stack)

//oefening 2
    vector<string> tabel[AANTAL];
    tabel[1].push_back("aap"); //nog geen elementen in de vector
    tabel[1].push_back("noot");
    tabel[1].push_back("mies");
    for(int i=0; i<AANTAL; i++){
        cout<<tabel[i];
    }

//oefening 3
    vector<vector<int> > v;
    for(int i=0; i<AANTAL; i++){
         vector<int> w(i);   //lengte gekend bij declaratie
        for(int k=0; k<v.size(); k++){
            w[k] = 10+10*k;
        }
        v.push_back(w);
    }
    //cout<<v;
    for(int i=v.size()-1; i>=0; i--){
        for(int k=v[i].size()-1; k>=0; k--){
            cout<<v[i][k]<<" ";
        }
        cout<<endl;
    }
    return 0;
}
```

**Oefening 114**

```
#include "containers.h"

int main(){
    map<char,unordered_set<string> > m;
    string woord;
    cout<<"geef woorden, eindig met STOP"<<endl;
    cin>>woord;
    while(woord!="STOP"){
        m[woord[0]].insert(woord);
        cin>>woord;
    }
    cout<<"geef een letter, ik zeg hoeveel verschillende woorden van
       daarnet met die letter starten ";
    char letter;
    cin>>letter;
    if(m.count(letter)==1){
        cout<<"er waren "<<m[letter].size()<<" verschillende woorden
           met die startletter"<<endl;
    }
    else{
        cout<<"er was geen enkel woord met die startletter"<<endl;
    }
    return 0;
}
```

---

**Oefening 115**

```
#include "containers.h"

int main(){
     vector<map<char,unordered_set<string> > > v(10); //vertrek van
        10 elementen
     string woord;
     cout<<"geef woorden, eindig met STOP"<<endl;
     cin>>woord;
     while(woord!="STOP"){
          if(woord.size() > v.size()-1){
               v.resize(woord.size()+1);
          }
          v[woord.size()][woord[0]].insert(woord);
          cin>>woord;
     }
     //cout<<v;

     //cout<<endl<<"geef een woord, ik zoek even lange woorden die
        met dezelfde letter starten ";
     //cin>>woord;
```

```
     woord = "sinterklaas";
     if(woord.size() < v.size()){
          if(v[woord.size()].count(woord[0])==1){
               cout<<v[woord.size()][woord[0]];
          }
          else{
               cout<<"geen woorden van die lengte met startletter
                  "<<woord[0]<<endl;
          }
     }
     else{
          cout<<"geen woorden gevonden die zo lang zijn"<<endl;
     }
     return 0;
}
```

---

**Oefening 116**

```
/*
(a) unordered_set<string>
(b) geen container nodig; meteen uitschrijven
(c) stack<string> of vector<string>
(d) geen container nodig; als je het woord op voorhand kent: gewoon
   tellen
(e) idem; meteen vindplaatsen uitschrijven
(f) map<string,int>
(g) map<string,vector<int> >
*/
```

---

**Oefening 117**

```
/* Waaraan kunnen wij zien of je een verkapte javaprogrammeur dan wel
   een echte c++-programmeur bent?
   Als je ergens "="-tekens hebt staan, gebruik je de toekennings-
   operator. En die TOEKENNINGSOPERATOR die MAAKT KOPIES in C++.
   En kopies maken is DUUUUUUUR.
   Dus dat doe je niet zonder geldige reden. */

#include "containers.h"

void vul_in_i_de_map_stack_bij_sleutel_aan_met_set_van_drie_strings(
         vector<map<string,stack<set<string>>>> & vect, int i,
         string sleutel, const string & str1, const string &str2,
         const string &str3){

    // OPDRACHT 1
    // een nieuwe set op stack die bij "noot" hoort steken:
    set<string> s;
```

```
    s.insert(str1);
    s.insert(str2);
    s.insert(str3);
    vect[i-1][sleutel].push(move(s));


    // IN C++11 OOK MOGELIJK OP 2 REGELS:
    set<string> st = {str1,str2,str3};
    vect[i-1][sleutel].push(move(st)); // liet je de 'move' weg?
                          // dan maak je nog een kopie van de set s
                          // (het gebruik van move (=std::move)
                          // komt ook nog glater in theorie aan bod)

    // ... EN ZELFS OP 1 REGEL,
    // waarbij je automatisch (zonder move) vermijdt om de set te
       kopieren:
    vect[i-1][sleutel].push({str1,str2,str3});
}

// wat je zeker NIET mag doen in vorige opdracht is kopieen maken
// FOUT!   map<string,stack<set<string>>> hulpmap = vect[i-1];
// FOUT!   stack<set<string>>  hulpstack = hulpmap[sleutel];
// etcetera

bool
   i_de_map_beeldt_woord_af_op_stack_waarvan_bovenste_set_dit_element_bevat(
       const vector<map<string,stack<set<string>>>> & vect, int i,
       const string & woord, const string & element){

    bool aanwezig = false;
    map<string,stack<set<string> > >::const_iterator it =
       vect[i].find(woord);
    if(it != vect[i].end()){ // woord is aanwezig
        if(!it->second.empty()){   // bijhorende stack is niet leeg
            //+/ hier opgelost zonder iterator
            aanwezig = (it->second.top().count(element)!=0);
        }
    }
    return aanwezig;
}

int main(){
    vector<map<string,stack<set<string> > > > v(5);

    vul_in_i_de_map_stack_bij_sleutel_aan_met_set_van_drie_strings(v,0,
            "noot","do","re","mi");
    cout<<v;

    if(i_de_map_beeldt_woord_af_op_stack_waarvan_bovenste_set_dit_element_bevat(
            v,0,"noot","re")){
```

```
        cout<<"\nmap op index 0 bevat sleutel 'noot', en element 're'"
            <<" zit in zijn bovenste set van de bijhorende stack";
    }
    else{
        cout<<"\nFOUT 1....";
    }

    if(i_de_map_beeldt_woord_af_op_stack_waarvan_bovenste_set_dit_element_bevat(
            v,0,"noot","sol")){
        cout<<"\nFOUT 2....";
    }
    if(i_de_map_beeldt_woord_af_op_stack_waarvan_bovenste_set_dit_element_bevat(
            v,0,"appelmoes","re")){
        cout<<"\nFOUT 3....";
    }
    if(i_de_map_beeldt_woord_af_op_stack_waarvan_bovenste_set_dit_element_bevat(
            v,1,"noot","re")){
        cout<<"\nFOUT 4....";
    }
    cout<<v;
    return 0;
}
```

---

**Oefening 118**

```
#include <iostream>
#include <set>
#include <string>
#include <ctime>
#include "containers.h"
using namespace std;

int main(){
     srand(time(0));
     int aantal = rand()%20+11;
     int max = 2*aantal;
     set<int> verzameling;
     while(verzameling.size()!=aantal){ //verschillende getallen !!
          verzameling.insert(rand()%max);
     }

    //  cout<<verzameling<<endl;
    set<int>::iterator it = verzameling.begin();
    int aantal = verzameling.size(); //zie opmerking
    for(int i=0; i < aantal ; i+=3){
   /* DIT WERKT, MAAR IS INEFFICIENT (omdat 'erase' weer op zoek moet
      gaan)
        int xx = *it;
        it++;
```

```
        verzameling.erase(xx);
        it++;
        it++;
   */
        set<string>::iterator it_hulp = it;
        it++;
        verzameling.erase(it_hulp); // (it-1) onmogelijk
        if(it!=verzameling.end()) it++; // anders is opschuiven niet
           netjes en kan tot runtime-fouten leiden
        if(it!=verzameling.end()) it++;
    }

    cout<<endl<<verzameling;

    return 0;
}

/* ZAKEN DIE NIET MOGELIJK / NIET JUIST ZIJN:

it = it+3;      (kan wel voor pointers, niet voor iteratoren)

erase(it) oproepen en hopen dat 'it' nog een zinvolle waarde heeft

testen op 'it != verzameling.end()' als je iterator 'it' telkens per 3
laat opschuiven; dan zal it in 2 van de 3 gevallen al VOORBIJ
'verzameling.end()' staan wijzen.

iteratoren vergelijken met < of <=

een for-lus gebruiken waarvan de eind-grens tussendoor wijzigt.
 for(int i = 0; i < verzameling.size(); i++){
      verzameling.erase(...);    --> WIJZIGT DE BOVENGRENS VAN DE LUS
      //...
 }
*/
```

---

**Oefening 119**

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



/****************** Versie zonder procedures ***********************/
```

```
int main(){
     map<int,string> tekst_op_regelnr;
     vector<int>volgorde_regelnummers;

     ifstream input("regelnummers.txt");
     if(!input.is_open()){
          cout << "bestand niet gevonden.... "<<endl;
     }
     else{
          int nr;
          while(input>>nr){
               tekst_op_regelnr[nr]=""; //nr toevoegen aan
                  sleutelverzameling
               volgorde_regelnummers.push_back(nr);
          }

          ifstream input2("nbible.txt");
          if(!input2.is_open()){
               cout << "bestand niet gevonden.... "<<endl;
          }
          else{
               int tel=1;
               string lijn;
               while(getline(input2,lijn)){
                    if ( tekst_op_regelnr.count(tel)>0){
                         tekst_op_regelnr[tel]=lijn;
                    }
                    tel++;
               }

               for(int i=0;i<volgorde_regelnummers.size();i++){
                    cout<<
                       tekst_op_regelnr[volgorde_regelnummers[i]]<<endl;
               }
          }
          return 0;
     }
}
//mogelijk alternatief: de getallen nog niet in de sleutelverzameling
   van de map steken, maar in een aparte set.

/****************** Versie met procedures ***************************/
void regelnummers_opslaan(map<int,string> & m, vector<int> & v,
          const string & bestand){
     ifstream input;
     input.open(bestand);
     if(!input.is_open()){
          cout << "bestand niet gevonden.... "<<endl;
     }
```

```
     else{
          int nr;
          while(input>>nr){
               m[nr]="";
               v.push_back(nr);
          }
     }
}

void tekstregels_opzoeken(map<int,string> & m, const string &
   bestand){
     ifstream input;
     input.open(bestand);
     if(!input.is_open()){
          cout << "bestand niet gevonden.... "<<endl;
     }
     else{  //(1) zie laatste opmerking
          string lijn;
          int nr=1;
          while(getline(input,lijn)){
               if(m.count(nr)==1){
                    m[nr]=lijn;
               }
               nr++;
          }
     }
}

//naar bestand weggeschreven
void regelnummers_vervangen(const map<int,string> & m, const
   vector<int> & v,
         const string & bestandsnaam){
     ofstream output(bestandsnaam.c_str());
     for(int i=0;i<v.size();i++){
          map<int,string>::const_iterator it=m.find(v[i]);
          output<<endl<<it->second;
     }
}

int main(){
     //regelnummers.txt bevat
        {18876,10000,27132,517,8999,29454,22002,2008,27312,25712}
     vector<int> volgorde_regelnummers;
     map<int,string> tekst_op_regelnr;
     regelnummers_opslaan(tekst_op_regelnr,volgorde_regelnummers,"regelnummers.txt"
     tekstregels_opzoeken(tekst_op_regelnr,"nbible.txt");
     regelnummers_vervangen(tekst_op_regelnr,volgorde_regelnummers,"verhaal.txt");
     cout<<"Het resultaat is te zien in het bestand
        'verhaal.txt'."<<endl;
     return 0;
```

```
}

// OPGELET! hieronder staat een alternatief voor de methode
   regelnummers_vervangen.
// In die procedure kan je echter niet de gewenste 'veiligheid'
   inbouwen zodat de map niet kan veranderd worden:
// hier kan geen const bij map staan, want de aanroep 'm[...]' zou
   eventueel iets kunnen
// wijzigen aan de map !! (zie compilermeldingen als je toch const
   toevoegt)
/*
  void regelnummers_vervangen(map<int,string> & m, const vector<int>
     & v, const string & bestandsnaam){
       ofstream output(bestandsnaam.c_str());
       for(int i=0;i<v.size();i++){
           output<<endl<<m[v[i]];
       }
  }
*/

/********************  Inlezen stoppen bij laatste lijn ************/
//Vervang de code in (1) door:

    int tel=1;
    string lijn;
    map<int,string>::iterator it = m.begin();
    while (it != m.end()){
         while(tel<=it->first){
              getline(input,lijn);
              tel++;
         }
         it->second=lijn;
         it++;
    }
```