UNIVERSITEIT GENT

FACULTEIT INGENIEURSWETENSCHAPPEN EN ARCHITECTUUR

LABO Programmeren in C en C++
Oefeningenbundel
PARTIM C++

Leen Brouns
Helga Naessens
Wim Van den Breen

Opleiding Industrieel Ingenieur Informatica / Elektronica / Automatisering

september 2017

Doelstelling van de labo's C en C++

Lees voor de doelstellingen van de labo's C++ de oefeningenbundel PARTIM C.

Software

Zelfde opmerking voor de software.

Instellingen Dev-C++

Voor C++ -programma's komt er onder de compileropties de optie `-std=c++11` (met een kleine c) en de linkeropties vink je aan.

REEKS A

Kennismaking met C++
cin / cout en aanverwante operatoren, (standaard)strings, templates, default parameters

Oefening 101

Herneem oefening 1, maar dan in C++: schrijf op het scherm

```
Hello world!
10 9 8 7 6 5 4 3 2 1
START
```

Oefening 102

Hier komt de C++ -variant van oefening 2. Schrijf een programma dat alle (gehele) getallen van 0 tot en met 64 uitschrijft. Per regel komt zowel octale, decimale, als hexadecimale voorstelling van één getal. Zorg ervoor dat de getallen rechts gealligneerd zijn. Dit kan aan de hand van de manipulator `setw(aantal)` die ervoor zorgt dat de volgende operand van de uitschrijfoperator `<<` een vaste breedte van `aantal` karakters inneemt. Te vinden in de bibliotheek `iomanip`.

Oefening 103

Voorspel zonder computer waar de compiler fouten zal bespeuren. Verwijder (enkel) die regels uit de code, en voer uit.

```
const int AANT=10;
int getallen[AANT];
char letters[AANT];
string woorden[AANT];
cout<<"Geef "<<AANT<<" getallen, "
    <<"letters resp. woorden: ";
cin>>getallen;
cin>>letters;
cin>>woorden;
cout<<getallen<<endl;
cout<<letters<<endl;
cout<<woorden<<endl;
```

Oefening 104

Stel dat je een karakterteken wil omzetten naar een string (van lengte 1). Argumenteer waarom welk stukje code (niet) werkt.

```
char c = 'x';
string s = "" + c;
cout << "karakter "<< c <<" omgezet: "<< s << "." << endl;

char k = 'y';
string w = "";
w += k;
cout << "karakter "<< k <<" omgezet: "<< w << "." << endl;
```

Oefening 105

1. Schrijf een functie `genereer_string(n)` die een random standaardstring genereert van lengte `n` (`n` is een gegeven geheel getal `>= 0`). De resulterende string bestaat dus uit een aaneenschakeling van n (al dan niet verschillende) random kleine letters.
2. Schrijf een procedure `vul_array_met_strings(tab, n, len)` die de array `tab` opvult met `n` random strings van lengte `len`. Je gebruikt hierbij uiteraard de voorgaande functie `geneer_string`.
3. Schrijf een procedure `bepaal_min_en_max(tab, n, min, max)` die in de gegeven array `tab` (die minstens `n` standaardstrings bevat) op zoek gaat naar de alfabetisch kleinste en grootste string (van de eerste `n`) en deze respectievelijk opslaat in de parameter `min` en `max`.
4. Test voorgaande procedures uit in een hoofdprogramma.

Oefening 106

Schrijf een functie `grootste(array,lengte)` die uit een gegeven array van gegeven lengte het grootste element teruggeeft. Het type van de elementen die in de array bewaard worden, is niet gekend. De grootte van een element wordt bepaald door de functie `grootte(elt)`, die dus zal geïmplementeerd moeten worden voor elk type waarvoor je de functie `grootste` oproept.

1. Schrijf een hoofdprogramma waarin je alvast volgende code schrijft:

   ```
   double getallen[5] = {5.5,7.7,2.2,9.9,9.8};
   string woorden[3] = {"geloof","hoop","de liefde"};
   cout << grootste(getallen,5) << endl;
   cout << "De grootste van de drie is " << grootste(woorden,3) << "." << endl;
   ```

   Zorg dat dit programma runt. De grootte van een woord wordt hier gedefinieerd als de lengte van het woord. Merk op dat we de functie `grootte(...)` niet meegeven aan de functie `grootste(...,...)`: we vragen hier dus geen functiepointers.
2. Vervolgens maak je ook een array van drie personen aan. Elke persoon is van type Persoon (definieer zelf de struct), en onthoudt zowel zijn naam als leeftijd (in jaren) en lengte (in meter). Schrijf een procedure `initialiseer(persoon,naam,leeftijd,lengte)` die de dataleden van een gegeven persoon initialiseert. Gebruik deze procedure om de drie personen in de array te initialiseren:
   De eerste persoon is Samuel, 12 jaar, 1m52.
   De tweede persoon is Jente, 22 jaar, 1m81.
   De derde persoon is Idris, 42 jaar, 1m73.
   Schrijf een procedure `print(persoon)` die de gegevens van een persoon uitschrijft op het scherm. (Test uit door een van de personen uit de tabel uit te schrijven.)
   Schrijf tenslotte de grootste persoon uit, als de grootte van een persoon bepaald wordt door zijn leeftijd. (Nadien pas je de code aan zodat de grootte van een persoon bepaald wordt door zijn/haar lengte respectievelijk de lengte van zijn/haar naam. Krijg je het verwachte resultaat?)

Oefening 107

Schrijf een procedure `schrijf(array,aantal,achterstevoren,tussenteken)` die een array van gehele getallen uitschrijft. De tweede parameter geeft de lengte van de array weer, de derde parameter bepaalt of het uitschrijven van achter naar voor dan wel op normale wijze moet gebeuren. De laatste parameter geeft het karakterteken mee dat tussen twee opeenvolgende getallen uitgeschreven wordt.

Declareer in het hoofdprogramma een array `t` met de getallen `1 3 5 7 9 11 13`. Roep daarna de procedure `schrijf` als volgt aan:

```
schrijf(t,7);
schrijf(t,7,true);
schrijf(t,7,false,'-');
schrijf(t,7,true,'-');
```

Dit zou de volgende output moeten produceren:

```
1 3 5 7 9 11 13
13 11 9 7 5 3 1
1-3-5-7-9-11-13
13-11-9-7-5-3-1
```

REEKS B

Kennismaking met unique pointers

Oefening 108

Gegeven onderstaande code.

```
#include <memory>
#include <iostream>
using namespace std;

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

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

int main(){

    string namen[]={"Rein","Ada","Eppo"};

    schrijf(namen,3);
    verwijder(namen,3,0);
    schrijf(namen,3);

    return 0;
}
```

Je weet dat de regel code `s[i] = s[i+1];` het kopiëren van een string impliceert. Dat moeten we vermijden, want een string kan in principe heel groot zijn. Schrijf twee nieuwe procedures, die bij het onderstaande hoofdprogramma horen. We bewaren nu (unique) pointers in de array, zodat we bij het opschuiven van de elementen in de array enkel pointers moeten verleggen, en geen kopieën maken.

```
int main(){
    unique_ptr<string> pnamen[]={make_unique<string>("Rein"),
                                 make_unique<string>("Ada"),
                                 make_unique<string>("Eppo")};
    schrijf(pnamen,3);
    verwijder(pnamen,3,0);
    schrijf(pnamen,3);

    return 0;
}
```

Probeer ook eens het laatste element uit de array te verwijderen!

Oefening 109

In oefening 22 gebruikte je een functie als parameter voor een andere functie of procedure. In C (en C++) moest je die functie eerst expliciet een naam geven en implementeren. In C++11 kan het in één moeite: als het functievoorschrift klein genoeg is, maak je de functie on-the-fly aan. Doe dit om onderstaand hoofdprogramma aan de praat te krijgen: gebruik λ-functies op de plaats van `.......`. De functie `vul_array` moet je ook schrijven. Let op, het type van de vierde parameter is nu geen functiepointer, want je werkt in C++11 in plaats van C!

```
int main(){
    int a[] = {0,1,2,3,4,5,6,7,8,9};
    int b[] = {0,10,20,30,40,50,60,70,80,90};
    int c[10];

    vul_array(a,b,c,...);
    schrijf("SOM:      ",c,10);
    vul_array(a,b,c,...);
    schrijf("PRODUCT:  ",c,10);
    vul_array(a,b,c,...);
    schrijf("VERSCHIL: ",c,10);

    return 0;
}
```

Oefening 110

Schrijf een C++-programma dat van alle (kleine) letters in het bestand `lord.txt` de frequenties uitschrijft. Om de frequenties bij te houden gebruik je een gewone array (nog geen vector); hoofdletters moet je niet tellen.

Oefening 111

Gegeven 2 tekstbestanden, `stationnetje.txt` en `paddestoel.txt`. Maak een derde bestand, `mix.txt`, waarin je de oneven regels van het eerste bestand laat afwisselen met de even regels van het tweede bestand. (Let wel: er zitten dus regels tussen die je *niet* in de mix terug zal vinden! Maak de mix zo lang als het kortste bestand.)

Breid dit daarna uit naar een mix van meerdere bestanden. Noem je programma `mix.cpp` en geef de bestandsnamen mee op de commandolijn. Het uitvoerbestand mag nog steeds `mix.txt` zijn. (Indien dit uitvoerbestand al bestaat, plak je de nieuwe output achteraan bij.)

De oproep kan er dan als volgt uitzien:

```
H:> mix stationnetje.txt paddestoel.txt khebdezonzienzakken.txt
```