Instellingen Dev-C++

Voor C++ -programma's komt er onder de compileropties de optie `-std=c++14` (met een kleine c) en bij linkeropties `-static-libgcc`. Beide opties vink je aan?

1 Basisconcepten C++/strings/templates/default param/bestanden

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

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

```
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 104

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`. Je mag veronderstellen dat de array `tab` groot genoeg is.
3. Schrijf een procedure `schrijf(tab,n)` die de gegeven array `tab` (die `n` standaard strings bevat) naar het scherm schrijft.
4. Schrijf een procedure `bepaal_min_en_max(tab, n, min, max)` die in de gegeven array `tab` (die `n` standaardstrings bevat) op zoek gaat naar de alfabetisch kleinste en grootste string en deze respectievelijk opslaat in de parameter `min` en `max`.
5. Test voorgaande procedures uit in een hoofdprogramma.

Oefening 105

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 getal is: "<< grootste(getallen,5) << endl;
cout << "Het grootste woord is: " << grootste(woorden,3) << "." << endl;
```

1. 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.
   Tip: Voeg een functie `grootte(elt)` toe. Deze functie bepaalt de grootte van een element, en moet geïmplementeerd moeten worden voor elk type waarvoor je de functie `grootste` oproept. De grootte van een woord wordt gedefinieerd als de lengte van het woord.
   Opmerking:
   De functie `grootte(...)` wordt **niet meegeven** aan de functie `grootste(...,...)`: we vragen hier dus geen functiepointers.
2. Maak in het hoofdprogramma een array van drie personen aan. Elke persoon is van type Persoon (definieer zelf de struct), en onthoudt zowel zijn naam (een `string`) als zijn leeftijd (een `int`) en lengte (in meter, dus een `double`).
3. 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 (vb. Samuel is 12 jaar, lengte is 1m52 / Jente is 22 jaar, lengte is 1m81 / Idris is 42 jaar, lengte 1m73).
4. Schrijf een procedure `print(persoon)` die de gegevens van een persoon uitschrijft op het scherm. (Test uit door een van de personen uit de array uit te schrijven.)
5. Schrijf tenslotte in het hoofdprogramma, 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 106

Schrijf een procedure `schrijf(array,aantal,achterstevoren,tussenteken)` die een array van gehele getallen uitschrijft. De tweede parameter bevat het aantal elementen in de array, 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
```

Oefening 107

Schrijf een C++-programma dat van alle (kleine) letters in het bestand `lord.txt` de frequenties uitschrijft (hoofdletters moet je niet tellen).

Om de frequenties bij te houden gebruik je een gewone array (nog geen vector);

Om het bestand in te lezen, gebruik je de juiste streams (geen input redirection).

De bestanden die je nodig hebt staan in de map **t**xt-bestanden op Minerva.

Oefening 108

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
```