LABO
Programmeren in C en C++
Oefeningenbundel

Leen Brouns
Helga Naessens
Wim Van den Breen

Opleiding Industrieel Ingenieur Informatica / Elektronica

september 2015

Doelstelling van de labo's C en C++

Met het aanbieden van de labo's C en C++ willen we inhoudelijk volgende zaken op een rijtje krijgen:

1. **inzicht in het verschil tussen C /C++ en Java**
   Zodat je op het juiste moment de juiste redenering en syntax gebruikt. Daarom zullen we je (voor dit ene vak) ook aanmoedigen om af en toe zonder IDE te werken: veel oefenen en zélf de code intikken laat je bewuster omgaan met de verschillen tussen Java en C /C++ .

2. **een reflexmatige voorkeur voor leesbare en efficiënte code**
   Je krijgt in deze cursus heel wat programmeeropgaven. Uiteraard gaan we er vanuit dat afgewerkte code de gevraagde output levert. Dat is echter nog maar de startvoorwaarde. Daarnaast is ook leesbaarheid en efficiëntie zeer belangrijk. Tip: vergelijk onze oplossingen altijd kritisch met je eigen code. Zit er een verschil in leesbaarheid en/of efficiëntie? Leer daarvan — of laat ons weten wat beter kan in de voorbeeldoplossing.

3. **een zekere vlotheid in je basishandelingen**
   Bepaalde programmaonderdelen, zoals aanbieden van een keuzemenu, (handmatig) zoeken in een tabel, bewerkingen op cijfers van gehele getallen,... komen dikwijls voor. Toch zeker als je met low-level-talen werkt. Even dikwijls zien we echter teveel ballast verschijnen in de geproduceerde code: teveel hulpvariabelen, overbodige testen, storende bewerkingen tussendoor. We geven je enkele stijltips, en sommen op welke basishandelingen je vlot uit je mouw moet kunnen schudden. (Dit staat uiteraard los van de gebruikte programmeertaal, maar we hebben hier de gelegenheid enkele van onze aandachtspunten aan jullie voor te stellen.)

4. **een goed begrip van het hot topic in C(++) : pointers**
   Dit komt jullie nog van pas in andere vakken (o.a. netwerkprogrammatie), daarom schakelen we in de oefeningen vrij snel pointers in.

5. **feeling met de 'low level' features in C**
   Hier denken we onder andere aan het gebruik van `char*` in plaats van `string`. Even doorbijten in het begin — het is écht niet gedateerd, en nog altijd nuttig.

6. **een gepast gebruik van de techniek van het recursief programmeren**
   (zowel in interface als functie-opbouw), met afweging van de voor- en nadelen ten opzichte van niet-recursief programmeren.

7. **een eerste overwinning op C++11**
   zodat jullie mee zijn met de recentste ontwikkelingen in het land van C++. Zie ook `http://fearlesscoder.blogsp`

8. ...

Meer algemeen streven we naar

1. **een denk- in plaats van tik-reflex**
   Vooral bij het gebruik van pointers, gelinkte lijsten, boomstructuren, pointers naar pointers,... is het van belang dat je de zaken al bij de eerste poging juist op papier zet. (Jawel, PAPIER, uitvinding van voor onze jaartelling, maar sedertdien o zo geduldig.) Begin je gehaast aan een kladpoging op je scherm, en sla je de bal of de pointer mis, dan kan elke kleine aanpassing (poging tot verbetering) je verder af drijven van het juiste pad. Dit straatje zonder eind kan zeer frustrerend werken, en 'tabula rasa' maken is dan dikwijls het enige aangewezen hulpmiddel.
   Is de oefening wat groter, en wil je aan alles tegelijk beginnen? Maak eerst een gerangschikt (!) to-do-lijstje dat je tijdens de loop van de opdracht kan afvinken. Dat vraagt soms dat je procedures of functies voorlopig 'schetst' (bvb. een hardgecodeerde waarde laat teruggeven, in afwachting van betere implementatie).

2. **propere manieren wat communicatie betreft**
   Hulp nodig tijdens een labo? Let op hoe je die vraagt. De zinsnede *Meneer/mevrouw, het werkt niet!* is geen vraag, maar een vaststelling waarop je lesgever enkel instemmend kan knikken. Zorg voor

   - een duidelijke omschrijving van de context
   - een stukje code
   - het probleem dat zich voordoet (compileerfout, error at runtime, onverwachte resultaten,...)
   - wat je eventueel zelf al probeerde

   We helpen je graag bij het juist formuleren van je vraag — dikwijls vind je hierdoor ook zelf al de oplossing.

3. **een goede inschatting van je eigen vorderingen en mogelijkheden**
   Voor de reguliere bachelorstudenten onder jullie: allicht hebben jullie nog een pak programmeerervaring op te doen. Op twee fronten tegelijk: Java én C(++) . Probeer gelijke tred te houden in beide vakken, en een speekmedaille af en toe kan alleen maar deugd doen!
   Voor de schakelstudenten: dit jaar wordt er veel van jullie verwacht — veel algemene vakken die na lange tijd weer op jullie agenda staan, ernstig voorbereid naar alle labo's komen, eventuele leemtes tussen vooropleiding en nieuwe leerstof zelf opvullen. Je zou voor minder het overzicht verliezen, of jezelf foutief inschatten.
   Voor beide groepen: jullie krijgen alvast enkele praktische hulpmiddelen — om te beginnen elke week een volledig uitgesponnen theorieles, in het labo een opeenvolging opdrachten van opbouwende moeilijkheidsgraad, geregelde feedback van de aanwezige docenten (al dan niet via testjes), oplossingen die je kritisch naast je eigen code dient te leggen. Doe je voordeel met de aangeboden hulp om zo vlot mogelijk de eindmeet te halen.

4. **een zicht op jullie kijk op de zaak**
   Aarzel dus niet om op- of aanmerkingen door te geven. IEDEREEN moet mee zijn met deze leerstof!
   Voor de schakelstudenten: ben je niet mee met deze leerstof, dan is er geen beginnen aan in het tweede semester (labo bij de cursus Algoritmen I).
   Voor de bachelorstudenten: jullie hebben bovendien nog een 'gap' van 12 maanden te overbruggen voor we aan de cursus Algoritmen I beginnen. Zorg dus dat de leerstof goed verankerd is!

Veel succes!

Software

Voor dit labo kan je kiezen: lokaal werken (dat kan op eigen laptop of op de labotoestellen in lokalen B2.030 tot B2.035) of via Athena. Zoek zelf uit wat voor jou het snelst/handigst werkt.

Via Athena

Lees aandachtig hoe je Athena best gebruikt op `www.helpdesk.ugent.be/athena/gebruik.php`. Zo kan je software gebruiken zonder die zelf te installeren.

Je beschikt over een Netwerkshare die je via Athena overal kan raadplegen (lees het onderdeel `Netwerkschijven`). Nadat Athena opgestart is kan je met `File Explorer` een verkenner openen vanuit Athena, waar je ook toegang hebt tot de lokale computer. Op deze share (de `H`-drive) staat alles veilig, er worden backups gemaakt, en je kan het van overal terug bereiken.

Toepassingen die je in Athena start werken veel sneller als je de bestanden ook opslaat op de netwerkdrive.

Open in het labo Athena zodat je alles kan uitproberen; voeg `Dev-C++` en `Command Shell` (beide onder `Academic/Development` te vinden) alvast toe aan `MyAthena`.

Maak op je `H`-drive een map aan voor deze cursus, en een submap voor de eerste reeks. Dan kan je starten. Gezien we in deze cursus vooral focussen op korte oefeningen, is het niet nodig om telkens een nieuw project aan te maken. Allicht heb je aan aparte bestanden genoeg.

Nog een weetje: allicht staan de instellingen bij de start op `qwerty`-klavier. Verander dit met `Alt-Shift`.

Lokaal

Wie kiest om lokaal te werken op de labotoestellen, moet weten dat de U-drive in onbruik is, en je dus de UGent-cloud moet mounten zodat je op je H-drive kan werken.

Wie op een eigen Windows-toestel werkt, downloadt de laatste versie van Orwell Dev-Cpp op `http://sourceforge.net/projects/orwelldevcpp/`.

REEKS 1

Kennismaking met C
main(), tabellen en eenvoudige functies / procedures

Oefening 1

Schrijf een programma dat volgende tekst op het scherm brengt (delay bij uitschrijven van de verschillende getallen is niet nodig). (En wat als we laten aftellen vanaf 100?)

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

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.

Oefening 3

Soms is het een goed idee om een nieuwe functie of procedure die je voor het eerst gebruikt, uit te testen in een kort programma. Zo kan je afchecken of je de functionaliteit van deze functie/procedure wel helemaal juist begrepen hebt.

Dit doen we hier voor de functie `scanf`. Het enige wat je moet doen, is heel aandachtig lezen wat de output wordt bij gegeven input.

Op Minerva vind je een .tex-bestand (`opg_16_03_chartabel_scanf.tex`) met de tex-versie van deze opgave. Download dit bestand, en kopieer het gegeven hoofdprogramma (dat je ook op volgende bladzijde vindt) in een .c-bestand. BRENG GEEN WIJZIGINGEN AAN! Introduceren van functies/procedures is hier NIET het topic van de oefening. Laat het programma lopen. Ga zeer aandachtig na wat de respons is van je programma op gegeven input. (Welke input dat is, bepaal je zelf.)

Je zou moeten merken dat het inlezen van verschillende types door elkaar niet altijd vlot verloopt. Tik je een letter waar een getal verwacht wordt, dan komt er onverwachte output (in plaats van een foutmelding), en wordt ook het vervolg van het inleesproces verstoord. Ook het inlezen van een karakter (waar witruimte wel als dusdanig ingelezen wordt, in plaats van overgeslagen) combineert niet goed met inlezen van woorden/getallen. Formuleer een uitleg voor elk vastgesteld gedrag. Maar onthoud vooral hoe `scanf` werkt met *woorden* en *getallen*. Vanaf nu zullen we `scanf`-opdrachten ook eenvoudig houden: niet teveel types gemixt.

```
#include <stdio.h>

main(){

    char woord [100];
    int geheelgetal;
    double kommagetal;
    char karakter;

    /* Voor elk type input (woorden, getallen, karakters) een aparte scanf-opdracht */
    printf("Geef een woord in: ");
    scanf("%s",woord);
    printf("Geef een geheel getal in: ");
    scanf("%i",&geheelgetal);
    printf("Geef een kommagetal in: ");
    scanf("%lf",&kommagetal);
    printf("Geef een karakter in: ");
    scanf("%c",&karakter);
    printf("Ik las in: %s|%i|%f|%c.\n",woord,geheelgetal,kommagetal,karakter);

    /* Woorden, getallen, karakters in dezelfde scanf-opdracht; spaties ertussen */
    printf("Geef woord, geheel getal, kommagetal en karakter in - ");
    printf("gescheiden door spaties.\n");
    scanf("%s %i %lf %c",woord,&geheelgetal,&kommagetal,&karakter);
    printf("Ik las in: %s|%i|%f|%c.\n",woord,geheelgetal,kommagetal,karakter);

    /* Woorden, getallen, karakters in dezelfde scanf-opdracht; komma's ertussen*/
    printf("Geef woord, geheel getal, kommagetal en karakter in - ");
    printf("gescheiden door komma's.\n");
    scanf("%s,%i,%lf,%c",woord,&geheelgetal,&kommagetal,&karakter);
    printf("Ik las in: %s|%i|%f|%c.\n",woord,geheelgetal,kommagetal,karakter);

}
```

Oefening 4

Schrijf een functie `faculteit(x)` die de faculteit van een gegeven geheel getal `x` berekent.
Schrijf een procedure `schrijf_faculteit(x)` die de faculteit van een gegeven geheel getal `x` uitschrijft.
Roep deze procedure op voor het getal 5. Daarna doe je dit voor alle getallen van 0 (0!=1) tot en met 40. Wat merk je? (Probleem dat zich stelt hoef je niet op te lossen, enkel te constateren.)

Oefening 5

Schrijf een functie `fibonacci(x)` die het Fibonacci-getal met volgnummer `x` berekent. De Fibonaccigetallen, startend bij volgnummer 1, zijn `1 1 2 3 5 8 13 21 34 55 ...` waarbij elk getal de som is van de twee vorige. (Behalve de twee eerste uiteraard, die zijn gewoon `1`.) Gebruik geen recursie.
Schrijf een functie `fibonacci_rec(x)` die hetzelfde doet, maar recursie gebruikt.
Schrijf aan de hand van deze functies alle Fibonacci-getallen van volgnummer 1 tot volgnummer 50 uit. Wat merk je?

Oefening 6

Wat doet deze code? Verklaar. Na theorieles 2 kan je de code zo omvormen, dat ze ook doet wat ze belooft.

```
void wissel(int a, int b){
     int hulp;
     printf("  Bij start van de wisselprocedure hebben we a=%i en b=%i.\n",a,b);
     hulp = a;
     a = b;
     b = hulp;
     printf("  Op het einde van de wisselprocedure hebben we a=%i en b=%i.\n",a,b);
}

int main(){
    int x, y;
    x = 5;
    y = 10;

    printf("Eerst hebben we x=%i en y=%i.\n",x,y);
    wissel(x,y);
    printf("Na de wissel hebben we x=%i en y=%i.\n",x,y);

    return 0;
}
```

Oefening 7

Zoek op hoe een computer of zakrekenmachine de sinus uitrekent van een gegeven getal. (Zeker weten dat-ie dat niet doet door een goniometrische cirkel te tekenen, en de ordinaat van het snijpunt van het ene been van de hoek met de cirkel af te passen!) Maak nu je eigen `mijn_sinus`-functie die van een gegeven kommagetal de sinus berekent. (Staat de gevraagde parameter in graden of radialen, denk je?) Opgelet: we kijken naar efficiëntie van je berekeningen! Vergelijk ook met het resultaat van de ingebouwde sinusfunctie uit de bibliotheek `math.h`.

Oefening 8

1. Schrijf een functie `cijfersom(x)` die van een gegeven geheel getal `x` de som van de cijfers berekent. Zo is `cijfersom(12345)` gelijk aan 15. Doe dit zonder bewerkingen op karaktertekens, gebruik enkel het type `int` (en wat wiskunde uit de lagere graad).
2. Gebruik deze functie in de functie `cijfersom_herhaald(x)`; deze blijft de som van de cijfers berekenen tot een getal kleiner dan 10 bekomen wordt. Zo is `cijfersom_herhaald(12345)` gelijk aan 6.
3. Maak nu een recursieve versie `cijfersom_rec(x)` die hetzelfde doet als `cijfersom_herhaald(x)`.

Oefening 9

In deze en volgende oefeningen gebruiken we een array. Omdat het doorgeven van een array aan een functie of procedure toch enige achtergrondinformatie vraagt, schrijven we hier geen functie/procedure maar enkel een hoofdprogramma. Houd je hier aan; later kan het anders.

Maak een array aan waarin elk element een karakter is. Vul deze array bij declaratie op met volgende letters: `b f r o a u v t o`. Wil je de lengte van een gegeven array kennen in C, dan ben je gedwongen volgende (lelijke) omweg te nemen: je vraagt de geheugenruimte die de array in beslag neemt (`sizeof(array)`), en deelt dit door de geheugenruimte die één element van de array in beslag neemt (in dit geval een `char`, dus `sizeof(char)`). Schrijf nu alle karakters uit die op een even positie in de array staan.

Oefening 10

Vul het volgende hoofdprogramma aan: het schrijft uit op welke plaats (=index, geteld vanaf 0) het ingelezen getal gevonden wordt.

```
#include <stdio.h>
main(){
    int rij []= {8,4,2,6,0,10};
    int x;
    printf("Geef een geheel getal op  ");
    scanf("%i",&x);

    ...


    if(...                    )
        printf("\nHet getal %i werd niet gevonden",x);
    }
    else{
        printf("\nHet getal %i werd gevonden op plaats %i.",x,...);
    }
}
```

Wat verandert er als je zeker weet dat de gegeven array al (stijgend) gerangschikt is? Test ook uit.

Oefening 11

Ook hier: nog geen functie of procedure, enkel een hoofdprogramma. Gegeven een hardgecodeerde array van karaktertekens. Vul het hoofdprogramma aan: schuif alle elementen in deze array één plaats naar links. Het eerste karakter komt achteraan. Als controle schrijf je de hele array uit. (Voorlopig nog geen functie, dat komt in volgende reeks aan bod.)

```
#include <stdio.h>
main(){
    char rij[] = {'s','a','p','p','e','l','m','o','e'};
    ...
```

Oefening 12

Nog een oefening met een array - dus voorlopig enkel een hoofdprogramma.
Gegeven een array met de coëfficiënten van een veelterm, waarbij de coeëfficiënt van de hoogste macht vooraan staat. De gebruiker geeft de waarde van de variabele x op. Bereken de functiewaarde van deze veelterm in de opgegeven variabele en schrijf uit.
Voorbeeld: de functiewaarde van $x^4 + 5x^2 - x + 7$ in $x = 12.1$, bekom je door de array de inhoud $\{1, 0, 5, -1, 7\}$ te geven. Het resultaat is 22162.838100.
Let op: je code moet zuinig zijn op bewerkingen!