
Programmeermethoden

Talen als Python; oude tentamens

Walter Kosters en Jonathan Vis

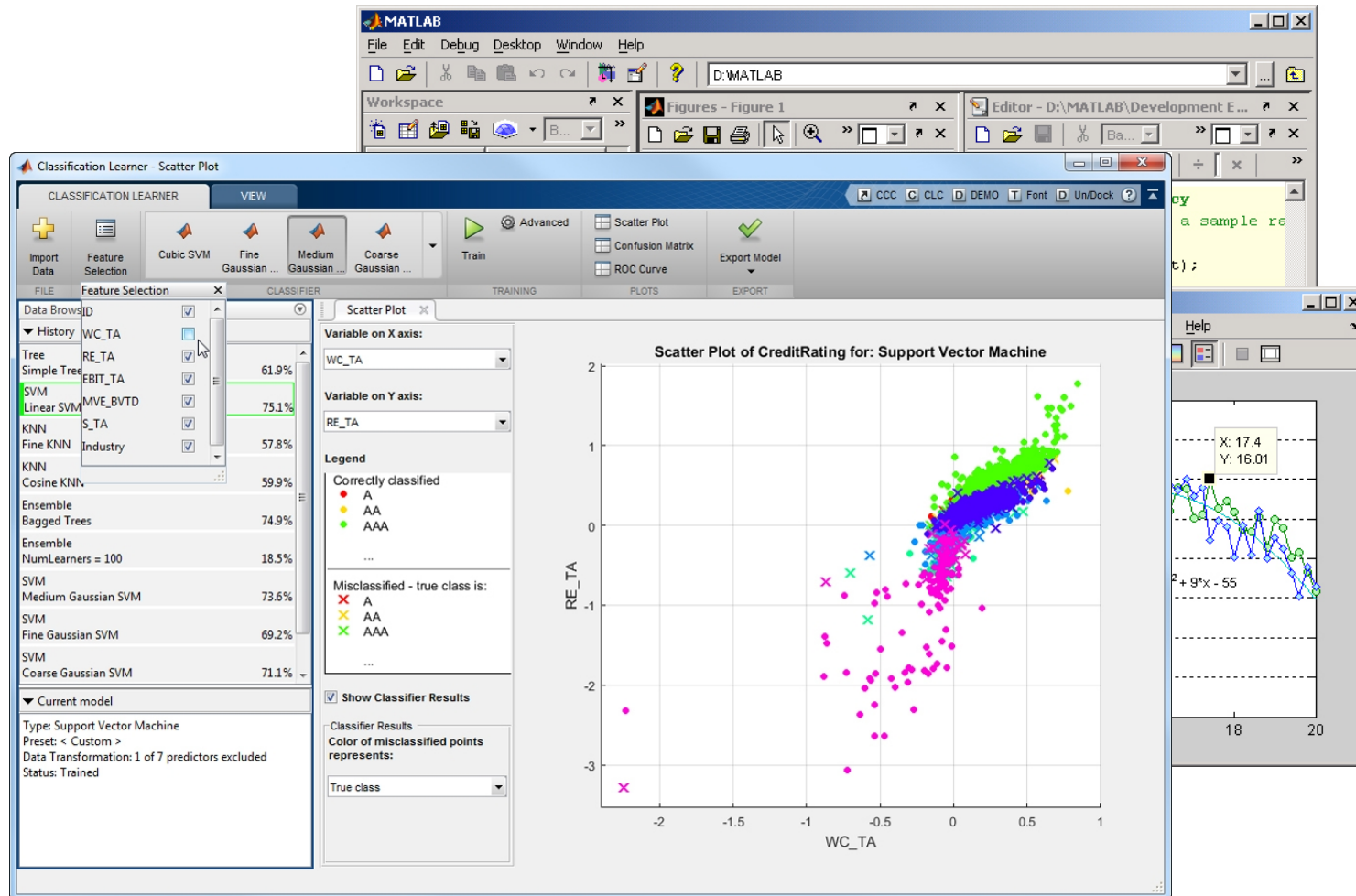
week 14: 13–17 december 2021

www.liacs.leidenuniv.nl/~kosterswa/pm/

Naast C++ (C: Brian Kernighan, Dennis Ritchie, Ken Thompson, \pm 1970; C++: Bjarne Stroustrup, \pm 1985, nu C++20), wordt ook veel **MATLAB** gebruikt, en **Java**. En **Qt** voor interfaces. En ... **Python**.



Ritchie en Thompson in 1972



www.mathworks.com

MATLAB, zelf in C geschreven, is met name goed in allerlei wiskundige operaties: matrices, ...; de taal heeft ook veel grafische mogelijkheden: plotten, ...

Nadeel: MATLAB is duur.

Naast de workspace heb je de “command history” en de “command window”.

Programma's (scripts) hebben extensie .m.

MATLAB is, net als Python, “weakly dynamically typed”.

De programmeertaal **Java** lijkt veel op C++.

Enkele belangrijke verschillen:

- Java is (nog) meer object-georiënteerd
- Java heeft automatische **garbage collection**
- Java heeft (bijna) geen pointers
- Java is platform-onafhankelijk
- Java kent **applets**: WWW-applicaties, met GUI's

Hoe compileer je een Java-programma, dat in Iets.java staat? Met `javac *.java` compileer je al je Java-files naar **bytecode**, in ons geval naar Iets.class. Deze kan met `java Iets` op elke computer gedraaid (om precies te zijn: *geïnterpreteerd*) worden met behulp van de **Java Virtual Machine (JVM)**. Dit bovenstaande geldt voor “tekst-applicaties”.

Een **applet** wordt bekeken met een webbrowser, en je moet dan een aparte HTML-pagina `naam.html` maken die de bytecode aanroept. Rechtstreeks (zonder browser) kan vaak ook eenvoudig met `appletviewer naam.html`, zie later.

Overigens, Java is iets heel anders dan JavaScript.

Maak een file Hello.java met

```
public class Hello {  
    public static void main (String[ ] argv) {  
        System.out.println ("Hello world!");  
    }//main  
}//class Hello
```



En dan `javac Hello.java` en `java Hello`.

Let op: filenaam = naam public klasse (met hoofdletter).

Er kunnen meer klassen in een file staan, maar er is er maar één `public`, en bij diens `public static void main` begint de executie van het programma.

```
import java.awt.*; import java.applet.*; import java.awt.event.*;
public class Muis extends Applet //Swing
    implements MouseListener, MouseMotionListener {
    public int teller = 0;
    public void init ( ) {
        addMouseListener (this); addMouseMotionListener (this); }//init
    public void paint (Graphics g) {
        g.setColor(Color.white); g.fillRect(10,10,500,500); }//paint
    public void mouseClicked (MouseEvent event) { }//mouseClicked
    public void mouseReleased (MouseEvent event) { }//mouseReleased
    public void mousePressed (MouseEvent event) { }//mousePressed
    public void mouseEntered (MouseEvent event) { }//mouseEntered
    public void mouseExited (MouseEvent event) { }//mouseExited
    public void mouseDragged (MouseEvent event) {
        Graphics g = getGraphics ( ); teller++;
        switch ( teller % 4 ) {
            case 0: g.setColor (Color.red); break;
            case 1: g.setColor (Color.green); break;
            case 2: g.setColor (Color.black); break;
            case 3: g.setColor (Color.blue); break; }//switch
        g.drawLine (event.getX ( ),event.getY ( ),
                    event.getX ( )+10,event.getY ( )+10); }//mouseDragged
    public void mouseMoved (MouseEvent event) { }//mouseMoved
}//Muis
```

www.liacs.leidenuniv.nl/~kosterswa/java/muis.html

www.liacs.leidenuniv.nl/~kosterswa/java/sleep.html

Wat we hier —en straks ook in Qt— zien is dat de “control-flow” **event-driven** (event-gestuurd) is. En dat is anders dan wat we gewend zijn!

Een onzichtbare hoofdloop (de “event-lus”) loopt “einde-loos” door. Intussen worden, door **events** zoals muis-acties af te handelen, de gebruikerswensen vervuld. Dat zijn precies methoden (= memberfuncties) behorend bij bepaalde objecten.

Qt (“cute” of “ku-tee”; de ‘t’ komt van toolkit) is een “cross-platform ontwikkelomgeving”, met een ingebouwde C++-bibliotheek. Samenwerken met Java kan ook. Voor niet commercieel gebruik is het gratis.

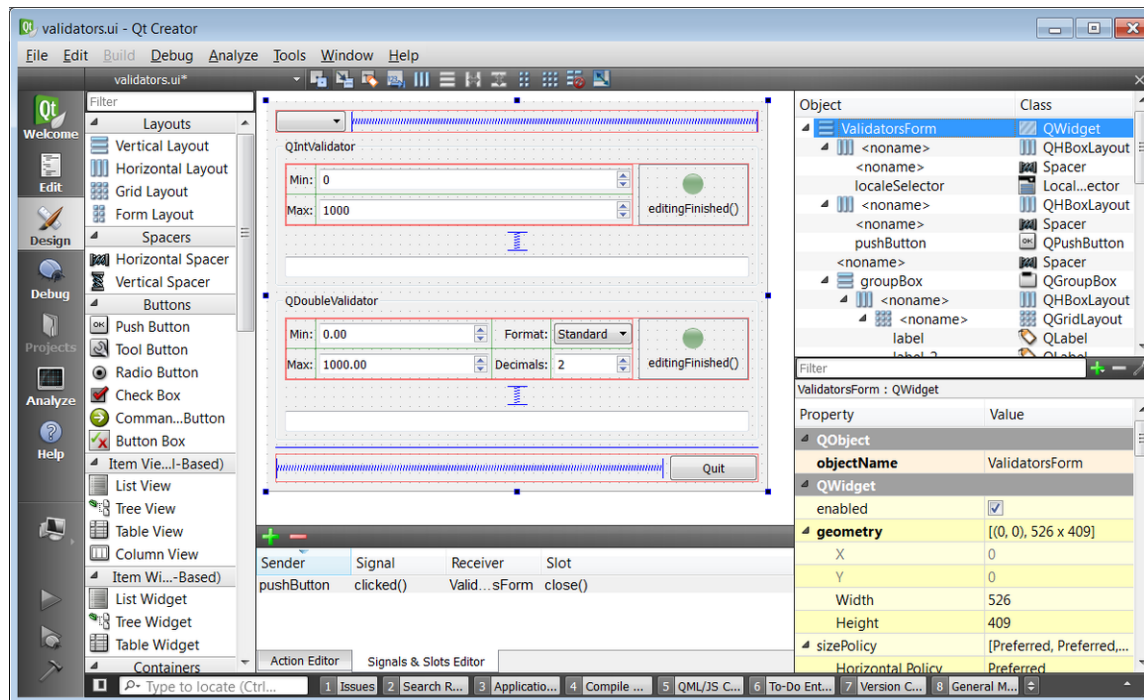


KDE, Google Earth, Mathematica en VLC zijn erin/mee geschreven.

Qt werkt met **signals** en **slots**. Als je iets doet, geeft je een signal, dat ergens door een slot wordt opgevangen.

Hoe leer je Qt? Of Qt Designer?

Ga naar een Linux-systeem. Geef in een terminal-window het commando `designer &` om Qt Designer te starten.



Maak vervolgens een mooie GUI (Grafische User Interface) voor Boter, kaas en eieren. Gebruik daarbij de C++-code in `boter.cc` die hier te vinden is:

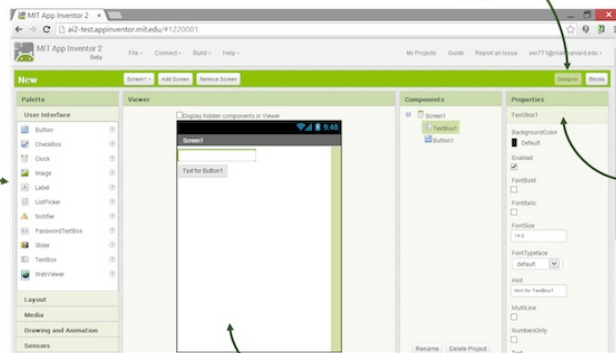
www.liacs.leidenuniv.nl/~kosterswa/pm/qt.php



En hoe maak je snel een **app**? Bijvoorbeeld met MIT APP Inventor, zie appinventor.mit.edu/

Palette: Find your components and drag them to the Viewer to add them to your app.

Designer Button: Click from any tab to go to the Designer tab.



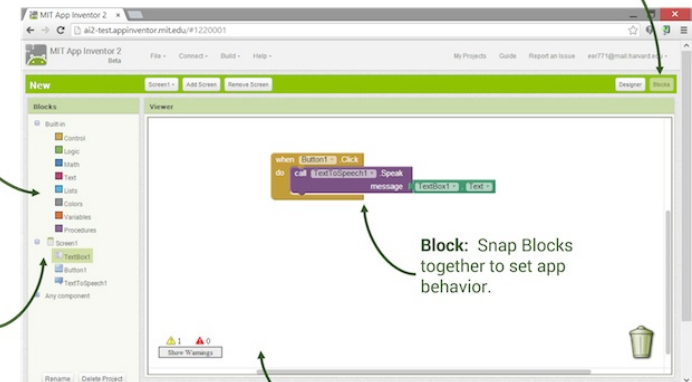
Properties: Select a Component in the Components List to change its properties (color, size, behavior) here.

Viewer: Drag components from the Palette to the Viewer to see what your app will look like.

Built-In Drawers: Find Blocks for general behaviors you may want to add to your app and drag them to the Blocks Viewer.

Blocks Button: Click from any tab to go to the Blocks tab.

Component-Specific Drawers: Find Blocks for behaviors for specific Components and drag them to the Blocks Viewer.



Block: Snap Blocks together to set app behavior.

Viewer: Drag Blocks from the Drawers to the Blocks Viewer to build relationships and behavior.

Python, ook zeer geschikt voor **scripting** en “prototyping”, is ontwikkeld door Guido van Rossum uit Nederland. Er zit veel overlap in met de andere genoemde talen.

Voor meer informatie (zelfstudie), zie:

www.liacs.leidenuniv.nl/~kosterswa/pm/pythonextra.php



- **interpreteren**, niet compileren: `python hello.py`

```
def hello():  
    print("Hello world!")  
  
hello()
```

- interactieve mode
- type van variabele kan eenvoudig wijzigen
- indentatie
- NumPy, Matplotlib, ...

```
# Dit is een regel met commentaar
import math # voor "pi"
print("Geef straal, en Enter .. ",end="")
straal = float(input())
if straal > 0:
    print("Oppervlakte: ",end="")
    print(math.pi * straal * straal)
else:
    print("Niet zo negatief ...")
print("Einde van dit programma.")
exit(0)
```

Geen accolades, geen punt-komma's. Dit is overigens **Python3**; er zijn subtiele verschillen met oudere versies.

Interactieve mode, via python3:

```
>>> a, b, c = 1, 7.1234, "een string" # gebruik " of '
>>> print(type(a),type(b),type(c))
<class 'int'> <class 'float'> <class 'str'>
>>> a = "wat anders"
>>> print(type(a),a)
<type 'str'> 'wat anders'
>>> 42**97 # machtsverheffen
285220783529230204153175149217567486191944489082385819
736594041463066207736137806080365924000128470355797067
70974141007921386304842702540555973307346454577152L
>>> a+d # gaat fout:
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'd' is not defined
>>> Ctrl-D
```

Let op de indentatie, oftewel het inspringen:

```
if y >= 3 and ( x == 4 or x == 5 ): # and voor && ...
    pass # doe niets
elif z == 12: # in plaats van else if uit C++
    x = 0
    y = 678
else:
    print("Hoe verzijn je het!")

for karakter in ['a', 'e', 'i', 'o', 'u']:
    print(karakter, end=" ")

som = 0
for i in range(6): # i = 0,1,2,3,4,5
    som = som + i*i
```

```
def telop(a,b):  
    c = a + b  
    return c  
  
q = telop(12345,6789)  
  
f = open("mijnfile.txt","r")  
for regel in f:  
    print(regel,end="")  
f.close()
```



Parameter-overdracht in Python is **call-by-object-reference**:

```
def vergroot(lijst):  
    lijst += [10,20,30,40]  
  
def vernieuw(lijst):  
    lijst = [1000,1001]  
  
lijst = [7,8]    # een "mutable" object  
vergroot(lijst)  
# nu is lijst [7,8,10,20,30,40]  
vernieuw(lijst)  
# en nu is lijst nog steeds [7,8,10,20,30,40]
```

En integers zijn “immutable”.

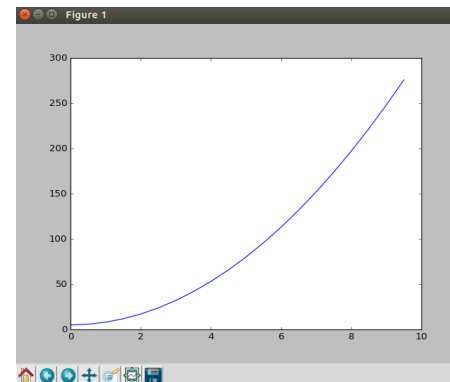
```
def simpelsort(A):
    for i in range(len(A)):
        # Zoek kleinste element in ongesorteerde stuk [i:]
        kl = i
        for j, el in enumerate(A[i:]):
            if el < A[kl]:
                kl = i + j
        # Wissel om
        if i != kl:
            A[i], A[kl] = A[kl], A[i]

# Test
B = [47, 10, 7, 3, 31, 75, 18, 21, 48, 79]
simpelsort(B)
print(B)
```

```
import numpy as np # hier zitten ook arrays in!
import matplotlib.pyplot as plt

# Bepaal de x-coördinaten waarvoor we willen plotten
x = np.arange(0, 10, 0.5)
# Bereken nu voor elk x-coördinaat de y-waarde
# Functie:  $y = 3x^2 + 5$ 
y = 3 * x * x + 5

# Geef de x- en y-arrays als parameters aan de plot-functie
plt.plot(x, y)
# Zet de plot op het scherm
plt.show()
exit(0)
```



Opgave 1 van het tentamen van 6 januari 2014:

In een array `int A[n]` staan `n` (een `const > 0`) gehele getallen.

a. Schrijf een C++-functie `hoevaak (A,X,n)` die teruggeeft hoe vaak het gehele getal `X` in het array `A` voorkomt.

b. Schrijf een Booleaanse C++-functie `uniek (A,n)` die precies dan `true` teruggeeft als geen enkel getal twee maal (of vaker) voorkomt in `A`, en anders `false`. Hierbij moet de functie van **a** *zinvol* gebruikt worden (hoe vaak komt `A[i]` voor?).

c. Schrijf een C++-functie `meest (A,n)` die het meest voorkomende getal uit `A` teruggeeft. Als er verschillende kandidaten zijn (bijvoorbeeld voor het array 17 12 30 12 42 30) moet het kleinste getal dat het meest voorkomt worden geretourneerd. In het voorbeeld is dit 12 (dat even vaak voorkomt als 30). Maak opnieuw gebruik van de functie van **a**.

d. Schrijf een C++-functie `sorteer (A,n)` die de getallen in `A` zodanig ordent dat voor alle getallen (behalve het laatste) geldt dat ze hooguit even vaak voorkomen als hun rechter buurman. Tip: pas de C++-code voor *bubblesort* eenvoudig aan; gebruik **a**.

e. Hoe vaak wordt de functie `hoevaak` aangeroepen in **d**?

soms dus ook iets over “complexiteit”: $O(n^2)$...

- ```
a. int hoevaak (int A[], int X, int n) {
 int i, teller = 0;
 for (i = 0; i < n; i++) if (X == A[i]) teller++;
 return teller;
}//hoevaak
```
- ```
b. bool uniek (int A[ ], int n) {
    int i;
    for ( i = 0; i < n; i++ )
        if ( hoevaak (A,A[i],n) > 1 ) return false;
    return true;
}//uniek
```
- ```
c. int meest (int A[], int n) {
 int i, tel, vaak = A[0], aantal = hoevaak (A,A[0],n);
 for (i = 1; i < n; i++) {
 tel = hoevaak (A,A[i],n);
 if (tel > aantal || (tel == aantal && A[i] < vaak)) {
 vaak = A[i]; aantal = tel; }//if
 }//for
 return vaak;
}//meest
```
- ```
d. void sorteer (int A[ ], int n) {
    int i, j, temp;
    for ( i = 1; i < n; i++ )
        for ( j = 0; j < n-i; j++ )
            if ( hoevaak (A,A[j],n) > hoevaak (A,A[j+1],n) ) {
                temp = A[j]; A[j] = A[j+1]; A[j+1] = temp; }//if
    }//sorteer
```
- e. $2 (1+2+\dots+n-1) = n (n-1)$ keer

Opgave 2 van het tentamen van 6 januari 2015:

a. Bij een functie kun je te maken hebben met *call by value* en *call by reference*, en ook met *locale* en *globale* variabelen. Verder onderscheiden we ook nog *formele* en *actuele* parameters. Leg deze zes begrippen duidelijk uit.

b. Gegeven een C++-programma met daarin de volgende twee functies:

```
int ludo (int a, int b, int n) {
    int i = 42; for ( i = 0; i < n; i += 2 ) { b += a; i--; }//for
    return b; }//ludo
int jeanine (int a, int b) {
    a = ludo (a,b,a); cout << a << "," << b << endl;
    a = ludo (a,b,a); cout << a << "," << b << endl;
    return a; }//jeanine
```

Verder zijn de globale variabelen *x* en *y* gegeven (van type *int*). Wat is dan de uitvoer van het volgende stukje programma (leg je antwoord duidelijk uit):

```
x = 2; y = 2; y = jeanine (x,y); cout << x << "," << y << endl;
```

c. Als **b**, maar nu met een *&* ("ampersand") bij de vijf parameters van de functies.

d. Geef een eenvoudige uitdrukking voor de return-waarde van een aanroep `ludo (a,b,n)`, uitgedrukt in diens parameters. Het maakt niet uit of er *&*'s bij de parameters staan.

e. Als **d**, maar nu voor `jeanine (a,b)`. Neem aan dat er bij alle vijf parameters een *&* staat, net als bij **c**.

f. Als in de functie `ludo` ergens `a = jeanine (a,2*n)`; staat, compileert het programma dan nog? Onderscheid gevallen met en zonder *&*.

"main"		"jeanine"		"ludo"				
x	y	a	b	a	b	n	i	
2	2	2	2	2	2	2	42	aanroep jeanine eerste aanroep ludo
					4		1	
		6		6			2	geeft 6 terug cout: 6,2
				6	2	6	42	tweede aanroep ludo
							0	
					8		1	
					⋮		⋮	
				38			6	geeft 38 terug cout: 38,2; geeft 38 terug
38		38						cout: 2,38

tijd ↓



"main"		"ludo"	tijd ↓
x	y	i	
= a uit jeanine = a uit ludo = n uit ludo	= b uit jeanine = b uit ludo		
2	2		aanroep jeanine eerste aanroep ludo
		42 0 1	
	4 6	2	geeft 6 terug cout: 6,6
6		42	tweede aanroep ludo
		0 1	
	12 ⋮	⋮	
	42	6	geeft 42 terug
42			cout: 42,42; geeft 42 terug
	42		cout: 42,42

d. $n \cdot a + b$ (als $n \geq 0$) e. $(a^2 + b)^2 + a^2 + b$

f. Dan moet de tweede parameter van `jeanine` call by value zijn, en een prototype van `jeanine` moet boven `ludo` staan.

Opgave 3 van het tentamen van 16 maart 2017:

Gegeven is het twee-dimensionale array `int afstand[n][n];`, met zekere `const int n ≥ 2`. Er geldt dat `afstand[i][j] > 0` de afstand is tussen de plaatsen `i` en `j` met `i ≠ j`, waarbij `afstand[i][i] = 0` is, en de afstand tussen `i` en `j` even groot is als die tussen `j` en `i`. Een voorbeeld met `n = 4` staat hiernaast.

0	3	7	9
3	0	4	14
7	4	0	8
9	14	8	0

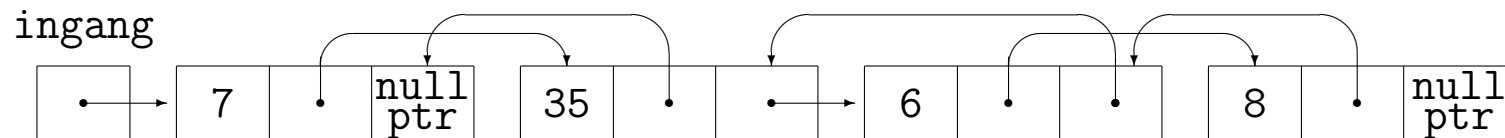
- a.** Schrijf een C++-functie `bool reis (afstand, km)` die kijkt of er een rondreis van `i` naar `j` naar `k` naar `i` is (voor willekeurige onderling verschillende plaatsen `i`, `j` en `k`) die in totaal precies afstand `km` heeft. In het voorbeeld zou voor 26 het antwoord `true` zijn: van 0 naar 1 naar 3 naar 0 (dat is $3 + 14 + 9 = 26$).
- b.** Schrijf een C++-functie `int verste (afstand, i)` die het nummer van de verst van `i` afgelegen plaats oplevert. In het voorbeeld, met `i = 3`, is dat 1 (wegens afstand 14). Als er meerdere plaatsen voldoen: die met de grootste index. Neem aan dat $0 \leq i < n$.
- c.** Schrijf een C++-functie `int hoever (afstand, i)` die bepaalt hoeveel je in totaal reist als je begint in `i`, dan steeds naar de verst gelegen plaats gaat, en stopt zodra je ergens komt waar je al eerder geweest was. In het voorbeeld, met `i = 0`, is het antwoord $9 + 14 + 14 = 37$ (van 0 naar 3 naar 1 naar 3). Neem aan dat $0 \leq i < n$. Hint: gebruik een Booleaans hulparray.

Opgave 4 van het tentamen van 3 januari 2019:

Gegeven is het volgende type:

```
class object { public: int info; object* volg1; object* volg2; };
```

Met behulp hiervan kan een lijst van objecten worden opgebouwd, bestaande uit vakjes met een getal, en twee pointers. Precies een van deze twee wijst naar het volgende object, de andere naar het vorige — maar je weet niet welke. Een voorbeeld, met `ingang` van type `object*`:



a. (6) Schrijf een C++-functie `voegtoe (ingang, getal)` die een nieuw object met `getal` erin netjes vooraan de lijst met `ingang` toevoegt. Je mag zelf kiezen welke van de twee pointers in het nieuwe object naar vorige en volgende object wijst. Zet in het oude eerste object (als dat bestaat) de terugwijzende pointer goed.

Opgave 4 van het tentamen van 3 januari 2019, vervolg:

- b.** (6) Schrijf een C++-functie `verwijder` (`ingang`) die het eerste object uit de lijst met `ingang` netjes verwijdert, indien dit bestaat.
- c.** (4) Schrijf een C++-functie `hoogop` (`ingang`) die als er minstens twee objecten zijn en als het `info`-veld van het eerste object oneven is, dit ophoogt met het `info`-veld van het tweede object. In het voorbeeld: 7 wordt 42.
- d.** (3) In de functies bij **a**, **b** en **c** staat in de heading een pointer. Deze heb je call by value of call by reference doorgegeven (met een `&`). Maakt het voor de werking van deze functies verschil uit of die `&` erbij staat? Mag het, moet het? Leg duidelijk uit.
- e.** (6) Schrijf een C++-functie `repareer` (`ingang`) die ervoor zorgt dat na afloop alle `volg1`-pointers naar het volgende, en alle `volg2`-pointers naar het vorige object wijzen.

www.liacs.leidenuniv.nl/~kosterswa/pm/tentamens.php

Werkcollege Programmeermethoden 16/17 december 2021

Tentamen 5 januari 2016

1. In een array `int A[n]` staan `n` (een `const ≥ 1`) gehele getallen > 0 .

a. (5) Schrijf een Booleaanse C++-functie `hoe(A, X, n)` die bepaalt of er een getal dat *hooguit* 1 van het gehele getal `X` verschilt in het array `A` voorkomt. Gebruik geen `(f)abs`.

b. (8) Schrijf een C++-functie `int langste(A, n, gem)` die de lengte uitrekent van een langste stijgende (of beter: niet-dalende) aaneengesloten deelrij in het array `A`. Hierbij moet `gem` het op de gebruikelijke manier naar een geheel getal afgeronde gemiddelde zijn van de getallen in de betreffende deelrij. Voor het array `1 7 1 1 2 7 6 3` zou het antwoord 4 zijn, namelijk de lengte van de deelrij `1 1 2 7`. En `gem` moet 3 worden (via $11/4$). Als er meer deelrijen dezelfde maximale lengte realiseren: het gemiddelde van de eerste.

c. (8) Schrijf een C++-functie `busort(A, n)` die het array `A` met de volgende variant van *bubblesort* oplopend sorteert. In de eerste ronde wordt het gehele array van links naar rechts doorlopen, in de tweede ronde van rechts naar links, in de derde van links naar rechts, etcetera. Stop als er tijdens een ronde geen verwisselingen waren.

d. (4) Hoeveel vergelijkingen tussen array-elementen worden minimaal/maximaal uitgevoerd in de functie van `c`? Geef voor beide situaties een voorbeeld.

2. (25) a. (6) Bij een functie kun je te maken hebben met *call by value* en *call by reference*, en ook met *local* en *global* variabelen. *Terse* of *directe* en *indirecte* ook nog *variabelen* en *actuele* parameters. Leg deze zes begrippen duidelijk uit.

b. (6) Gegeven een C++-programma met daarin de volgende twee functies:

```
int hillary(int n, int m) {
    n--; return n+m-1; } //hillary
int donald(int n, int m) {
    int a = 6; m--; n += 2; a++;
    m = n + hillary(n, m) + hillary(m, n) + a;
    cout << a << ", " << m << ", " << n << endl; return a+n-m; } //donald
```

Verder zijn de globale variabelen `a` en `m` gegeven, beide van type `int`. Wat is dan de uitvoer van het volgende stukje programma (leg je antwoord duidelijk uit):

```
a = 1; m = 4; cout << donald(m, a) << endl;
cout << a << ", " << m << endl;
```

c. (5) We voegen nu vier maal een `&` toe bij de parameters in de heading van `hillary` en `donald`. Beantwoord opnieuw vraag **b**; leg uit waarom verschillende uitkomsten mogelijk zijn, en geef deze.

d. (4) Als in de functie `hillary` ergens `a = donald(42, a-a)`; staat, compileert het programma dan nog? Onderscheid gevallen met en zonder `&`.

e. (4) Geef een eenvoudige uitdrukking voor de functiewaarde van `donald(r, s)`, uitgedrukt in `r` en `s`, voor de situatie zonder `&`'s (net zoals bij **b**).

3. (25) Gegeven is een `m` bij `n` (beide `const > 0`) array `M` met gehele getallen ≥ 0 . Hierbij geeft `M[i][j] ≥ 0` het aantal mensen ter plaatse (i, j) aan (met $0 \leq i < m$ en $0 \leq j < n$). Zie hiernaast voor een voorbeeld met `m = 4` en `n = 5`. De constanten `m` en `n` hoeven bij deze opgave niet doorgegeven te worden als parameter.

3	0	4	2	0
7	2	4	4	2
1	2	0	4	1
6	9	3	1	0

a. (7) Schrijf een C++-functie `int druk(M, som)` die de index van de drukste kolom, dat wil zeggen de kolom met de grootste som, geeft. In `som` moet de desbetreffende som komen. In het voorbeeld kolom 0, met `som` gelijk aan 17. Als er meerdere kolommen dit maximum realiseren, geef degene met de hoogste kolomindex.

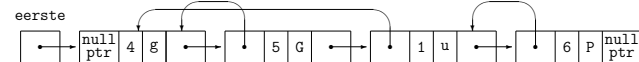
b. (8) Schrijf een C++-functie `int duos(M)` die het aantal horizontaal of verticaal direct aan elkaar grenzende (eventueel overlappende) paren geeft, waarbij de een precies het dubbele van de ander is. In het voorbeeld: 4-2, 2-4, 4-2, 1-2, 2-4, 2-1: 6 stuks.

c. (10) We lopen als volgt door het array `M`. Start bij (i, j) . De waarde van dit array-element geeft aan hoeveel plaatsen naar rechts je gaat; de waarde van de nieuwe plek bepaalt hoeveel plaatsen je omlaag gaat, enzovoorts. Als je rechts het array uitloopt kom je in dezelfde rij links weer binnen, en analoog voor de verticale richting. De wandeling stopt als je ergens komt waar je al eerder bent geweest (in het bijzonder bij een verplaatsing van 0). Schrijf een C++-functie `int aantal(M, i, j)` die het aantal stappen uitrekent; in `i` en `j` moeten de coördinaten van het laatst bezochte punt komen. In het voorbeeld, beginnend bij $(0, 0)$ eerst 3 naar rechts, dan 2 omlaag, dan 4 naar rechts, dan 0 omlaag en klaar: antwoord 4, en $(i, j) = (2, 2)$. Hint: gebruik een Booleaans hulpparray.

4. (25) Gegeven is het volgende type:

```
class kamer { public: kamer* vorig; int nr; char naam; kamer* volg; };
```

Een `kamer` object van type `kamer` wordt gemaakt (na `m` een klein letter of hoofdletter). Het veld `vorig` bevat een pointer naar het volgende `kamer` object, en `vorig` naar het vorige of voor-vorige, of `nullptr` als die er beide niet zijn. Een voorbeeld (eerste van type `kamer*`):



a. (5) Schrijf een C++-functie `voegtoe(eerste, kamernr, kamernm)` die een nieuw `kamer`-object met `kamernr` en `kamernm` erin vooraan in de lijst met ingang `eerste` toevoegt. Zet de `vorig`-pointer van het oude voorste object (als dat bestond) ook goed.

b. (5) Schrijf een C++-functie `verwijder(eerste)` die het voorste `kamer`-object uit de structuur die door `eerste` wordt aangewezen, netjes verwijdert — mits het bestaat. Zet eventuele `vorig`-pointers die er naar wijzen goed.

c. (5) Schrijf een C++-functie `wissel(eerste)` die de `kamer`-nummers van de twee voorste kamers omwisselt, mits de ene letter de hoofdletter van de andere is (zoals in het voorbeeld; 4 en 5 worden verwisseld). Controleer of de lijst minstens twee objecten heeft.

d. (4) In de functies bij **a**, **b** en **c** staat in de heading een pointer. Deze heb je *call by value* of *call by reference* doorgegeven (met een `&`). Maakt het voor de werking van deze functies verschil uit of die `&` erbij staat? Mag het, moet het? Leg duidelijk uit.

e. (6) Schrijf een C++-functie `herstel(eerste)` die alle `vorig`-pointers, behalve die van de voorste twee objecten (zo die al bestaan), naar het voor-vorige object laat wijzen. In het voorbeeld zou de pointer bij het object met `P` erin naar het object met `G` erin moeten gaan wijzen, verder verandert er niets.

Zie www.liacs.leidenuniv.nl/~kosterswa/pm/tentamens.php voor meer vragen en antwoorden!

Opgave 2 van het tentamen van 4 januari 2013:

b. Gegeven een C++-programma met daarin de volgende twee functies:

```
bool mark (int a, int b) {
    int z;
    a = a + b; b = a - b; a = a - b; cout << "M" << a << ", " << b << endl;
    z += 10; return ( a < b );
} //mark
int diederik (int b, int a) {
    bool temp; if ( mark (b,a) ) a += 2;
    while ( a > 0 ) { temp = mark (a,b); a--; cout << a << ", " << b << endl; }
    z += 10; return ( a + b + 2 );
} //diederik
```

Verder zijn de globale variabelen `x`, `y` en `z` gegeven (van type `int`). Wat is dan de uitvoer van het volgende stukje programma (leg je antwoord duidelijk uit; tip: 9 komma's):

```
x = 1; y = 3; z = 1; x = diederik (y,z); cout << x << ", " << y << ", " << z;
```

c. Als **b**, maar nu met een `&` ("ampersand") bij de vier parameters van de functies.

d. Als **c**, dus met vier `&`'s erbij, voor:

```
x = 1; y = 3; z = 1; y = diederik (y,y); cout << x << ", " << y << ", " << z;
```

e. Als in `mark` ergens `a = diederik (static_cast<int>(mark (a,b)),y)`; staat, compileert het programma dan nog? Onderscheid gevallen met en zonder `&`.

"main"			"diederik"			"mark"		tijd ↓
x	y	z	a	b	temp	a	b	
1	3	1	1	3	?			aanroep diederik
						3	1	eerste aanroep mark (z doet er niet toe)
						1	3	cout: M1,3; geeft true terug
			3					"a += 2;" uit diederik
						3	3	tweede aanroep mark
						3	3	cout: M3,3; geeft false terug
			2		false			cout: 2,3
						2	3	derde aanroep mark
						3	2	cout: M3,2; geeft false terug
			1		false			cout: 1,3
						1	3	vierde aanroep mark
						3	1	cout: M3,1; geeft false terug
			0		false			cout: 0,3
		11						geeft 5 terug aan x
5								cout: 5,3,11

"main"			"mark"	tijd ↓
x	y	z	temp	
	= b uit diederik	= a uit diederik		
1	3	1	?	aanroep diederik
	= a uit mark	= b uit mark		eerste aanroep mark
	1	3		cout: M1,3; geeft true terug
		5		"a += 2;" uit diederik
	= b uit mark	= a uit mark		tweede aanroep mark
	5	1		cout: M1,5; geeft true terug
		0	true	cout: 0,5
		10		geeft 17 terug aan x
17				cout: 17,5,10

"main"			"mark"	
x	y	z	temp	tijd ↓
	= a uit diederik = b uit diederik			
1	3	1	?	aanroep diederik aanroep mark
	= a uit mark = b uit mark 0	11		cout: M0,0; geeft false terug geeft 2 terug aan y cout: 1,2,11
	2			

e. Dit mag alleen als er geen & staat voor de eerste parameter van diederik, en er boven mark een prototype van diederik wordt gezet.

- werk wellicht nog aan de vierde programmeeropgave — de deadline is al voorbij!
- werkcollege 16/17 december 2021: oude tentamens
- **tentamen:**
woensdag 5 januari 2022, 14:15–17:15 uur, Gorlaeus zalen 2 (extra tijd) en 4/5
- **hertentamen:**
donderdag 31 maart 2022, 14:15–17:15 uur, Gorlaeus zalen 1 en 2
- www.liacs.leidenuniv.nl/~kosterswa/pm/