Tuyaux 1ste Bachelor Informatica : 1ste semester 2011-2012

WINAK Mentor Informatica Glenn De Jonghe

31 oktober 2011

Opmerkingen

Inleiding Programmeren & Inleiding tot C++ zijn de oude vakken en zijn slechts ter verwijzing toegevoegd onderaan het document.

Inhoud

Omdat ik de inhoud rechtstreeks van de wiki heb gehaald, zijn de paginanummers per vak. Hieronder de volgorde van de vakken waarin ze voorkomen in dit document.

Computersystemen en -architectuur

Uit Encyclopedia Academia

Professor Vangheluwe

Computersystemen en -architectuur	
Professor	Hans Vangheluwe
Richting	Informatica
Jaar	1BINF

Theorie

Het theorie gedeelte van het examen is simpele vraag & antwoord op het niveau van de middelbare school. Er wordt niet veel inzicht in de leerstof verwacht en alles staat haast letterlijk in de cursus. Het is aangeraden om het boek te lezen, en voornamelijk te leren van de slides aangevuld met persoonlijke nota's. Elk onderdeel van de theorie wordt getoetst, dus leer zeker ook alle verschillende representaties van getallen en hoe de onderlinge conversies juist gebeuren.

Praktijk

Het praktijkexamen bestaat erin om een skelet datapath verder uit te bouwen. Er zijn drie à vier opgaven die gegeven worden, dus als je in de groep zit die het laatst het examen krijgt kan je je opgaven al op voorhand te weten komen. Het datapath dat gemaakt moet worden is een simpelere versie van die je tijdens het jaar moet maken, met een beperkte instructieset.

Puntenverdeling

Meer dan de helft van de punten staat op dagelijks werk dat in groepjes van twee moet ingediend worden. De rest van de punten wordt bepaald tijdens de examenperiode. Er is ook een mondelinge verdediging van de projecten uit het dagelijks werk, zorg dus dat je goed snapt hoe alle opdrachten die je groepje heeft ingediend werkt.

Examenvragen

Academiejaar 2010 - 2011 - 1ste zittijd

Praktijk

Gegeven is een volledig uitgewerkte Program Counter/Register File en ALU. In het geheugen is plaats voor 1024 instructies. De RF bevat 8 registers, per conventie worden deze zo benut. (Noot: Registers worden aangeduid door notatie Ri en Rj.)

- R0 – R3 : Tijdelijke registers ($t0-$t3)
- R4 : Return Adress ($ra)
- R5 : Stack Pointer ($sp)
- R6 : $v0 (analogie met mips)
- R7 : Argument Register ($a0)

Je datapad moet de volgende instructies kunnen uitvoeren.

- **OR** Ri, Rj # Ri = Ri | Rj
- **ADD** Ri, Rj # Ri = Ri+Rj
- **SUB** Ri, Rj # Ri = Ri - Rj

- **BLT** Ri, Rj, offset # Als Ri kleiner is dan Rj, wordt er gebrancht naar PC+1+offset
- **SW** Ri, Rj, offset # De waarde van Ri wordt gestored op adress : Rj+offset
- **LD** Ri, Rj, offset # Laad in Ri, Rj+offset
- **LDI** Ri, immediate # Laad in Ri, een getal in
- **JR** Ri, offset # Jump naar Ri+offset
- **JAL** <8bit-adress> # Jump naar <adres> EN save het huidige adres in R4

De instructies hebben volgende binaire voorstelling:

- OR : 000iiijjj000
- ADD : 000iiijjj001
- SUB : 000iiijjj010
- BLT : 001iiijjjooo
- SW : 010iiijjjooo
- LD : 011iiijjjooo
- LDI : 100iiimmmmmm
- JR : 101iiiooooo
- JAL : 110xaaaaaaaa

- ooo : 3-bit 2's complement offset.
- Iiijjj : Met iii en jjj respectievelijk de unsigned voorstelling van het registernummer (0-7)
- mmmmmm : Unsigned Immediate
- aaaaaaaa : Unsigned Adress

Dit wetende, implementeer een 12-Bit datapad dat de gegeven instructieset ondersteunt.

Vragen

- Hoe zou je de pseudo instructie NOOP (NO Operation) implementeren dmv bestaande instructies te gebruiken? (noot: meerdere oplossingen mogelijk)
- Hoe zou je de pseudoinstructie CP (Copy Ri, Rj # kopieer de waarde van Rj, naar Ri) implementeren door gebruik te maken van de bestaande instructies?
- Hoeveel extra instructies kunnen aan deze instructieset worden toegevoegd door de huidige binaire vorm te behouden?
- Program Counter
  - Is de program counter, Signed of Unsigned?
  - Hoeveel bits moet de PC groot zijn?
  - Hoe groot moet het Instructie geheugen zijn (in bytes)?

Divers

Schrijf een programma dat de volgende code in C voorstelt door gebruik te maken van de gegeven instructies. (noot: Je kan geen gebruik maken van labels, adressen dienen door hun numerieke waarden opgeroepen worden.)

```
{value = 2
push(value)
value = value+i
pop(value)
}
```

Schrijf een eenvoudig programma dat de volledige functionaliteit van je Datapad toont.

Professor D'Haene

Bespreking

Theorie

Op zich is dit vak vrij eenvoudig: de cursus is niet erg uitgebreid, de leerstof is op zich niet zo moeilijk, noch is het examen dit. Het is echter een feit dat er niet met punten wordt gegooid en dat dit vak vaak wordt onderschat zodat niet iedereen slaagt.

Groso modo probeert dit vak een inleiding te geven in de werking van computersystemen. De theorie van dit vak is onderverdeeld in de gewone theorie waarvan de cursus bestaat uit slides en dan de theorie over besturingsystemen.

De gewone theorie wordt gegeven door professor D' Haene zijn dus dan zullen vele zaken van deze bespreking niet meer gelden) en omschrijft de inwendige bouw van computers, geschiedenis van de computers, inleiding electronica, inleiding Floating-Point getallen, telsystemen. Op zich is dit alles interessant, ware het niet dat de prof dezelfde slides als in de cursus gebruikt voor zijn lessen en deze nogal letterlijk afleest voor de studenten (in het vak Computerarchitectuur verbetert dit wel sterk aangezien deze stof interessanter is voor hem). Elke week zag je dan ook het aantal studenten die de lessen bijwoonden dalen. Kortom, het is dus niet ECHT nodig om naar de les te komen hier (al is het wel handig).

Het theoretisch examen telt vrij veel vragen zodat, als je een bepaalde vraag niet goed weet, je dit nog kan ophalen. Denk eraan voldoende uitleg te geven overal en zie dat je de flip-flop goed kent aangezien dit altijd gevraagd wordt en op veel punten gaat. Het mondeling examen is bij een goed schriftelijk examen een rustig gesprek waarbij je met een beetje geluk naar een kennisdemonstratie van D'Haene kan luisteren en hij eigenlijk meer spreekt dan jij.

Over de theorie van de besturingssystemen weet ik zelf niet echt hoe dit in zijn werk ging dit jaar.

Praktijk

De praktijk is dan weer wel interessant om bij te wonen, de sfeer is er luchtig en de assistent (Kurt Vanmechelen) is tof om mee te werken. In het begin kan de materie van de praktijk wat vaag overkomen (machinetaal, omzetten talsystemen), maar door samen te werken wordt dit al snel een routinezaak.

Het praktijkexamen van dit gedeelte bevat geen echte verrassingen en zijn varianten op behandelde zaken tijdens de oefeningensessies. Zie wel dat je het gedeelte over machinetaal goed begrijpt.

De praktijk over de Operating Systems bestaat erin in Solaris wat BASH-scripts te schrijven onder leiding van David Dewolfs om kennis te maken met UNIX. Dit gedeelte gaat maar op 3 punten van het totaalpakket, op zich niet zo veel, maar omdat deze introductie voor velen een eerste kennismaking is met het UNIX-besturingssysteem kan je beter komen.

Het examengedeelte van dit laatste bestaat er dan ook uit enkele ietwat complexere BASH-scripts te schrijven.

Puntenverdeling

Computersystemen: 13/20 (waarvan 7/20 op theorie en 6/20 op praktijk), besturingssystemen op

7/20 (waarvan 3/20 op praktijk en 4/20 op theorie). Dit kan echter afgelopen jaar zijn gewijzigd.

Examenvragen

Academiejaar 2007 - 2008 - 1ste zittijd

Theorie

1. Verklaar de volgende afkortingen:
   - ISP
   - ISA
   - PMT
   - IP
   - RGB
   - FCFS
   - IEEE
   - EBCDIC
2. Leg uit: CPU scheduling algoritmes.
3. Teken en leg uit : Stroomdeler.
4. Leg uit de manieren van VideoCompressie.
5. Leg uit: Non-Preemtive en Preemtive scheduling.
6. Leg uit, Fetch , decode, execute Cycle.
7. Invertor maken met transistoren.
8. Nyquist en Shanon Criteria.
9. DSL (uitgebreid + mondeling toelichten).
10. Disk - Scheduling algoritmes (uitgebreid + mondeling toelichten).
11. SR-latch met NOR Poorten (uitgebreid + mondeling toelichten).

Praktijk

1. Do the following conversions:
   1. (1210212) in basis 3 naar basis 9.
   2. (110110) in basis 2 (6-bit 1's complement) = (?) in basis 2 excess 32.
2. Calculate the sum
   1. (1A)basis 12 + (13.3)basis 4 = (?)basis 2
3. We consider a floating point representation with a sign bit, followed by an excess-15 exponent of 5 bits, followed by a normalized 8-bit fraction with a hidden bit. The comma is located behind the first non-zero digit. Numbers that cannot be represented exactly are rounded to the nearest number that can. 0 is represented by the reserved bit pattern 000000000000000. The exponents 00000 en 111111 are reserved for special cases and cannot be used for normal number representations.
   1. What is the number of representable numbers?
   2. (-0.123)basis 4 to this floating-point representation.
4. Construct a truth table for a function that calculates the sum of 3 - 1 bit input signals A1 ; A2 ; A3 . The output is the sum of the bits that consists S1,S2 (vb A1 = 1 en A2 = 0 en A3 = 1 => S1 = 1 S2 = 0). Implement also with 2 8-to-1 multiplexers *(Grappige noot:*

oorspronkelijk zou de oefening met een 4-to-1 multiplexer zijn geweest maar een kleine typo was in het examen geslopen).

5. Write Pep/7 assembly code that reads a number between 1 and 26 as input, and prints the respective character to screen. For Ex if the input is 5 , the output should be E and if the input is 26 , the output should be Z.

6. What is the content of the accumulator (A) after execution of the following code?

    LOADA d#0,i LOADX Z,d

    X : STOREX X,d Y: ADDA X,d SUB d#1,i BRGT Y STOP Z : .WORD d#5 .END

Academiejaar 2000 - 2001 - 1ste zittijd

Theorie

1. Noem de 5 elementen van een computer volgens het Von Neumann model.
2. Hoeveel megahertz zijn er in 1 gigahertz?
3. Is het mogelijk om elk decimaal nummer exact voor te stellen als een binair nummer?
4. Schrijf het binair getal $(101101011101)_2$ in het 8-deling getallenstelsel $(octaal)=(.....)_8$

Praktijk (compilatie van meerdere jaren)

1. Kan je de binaire getallen gemakkelijk omzetten naar hexadecimale getallen door het binair op te delen in groep jes van 3 binairen?
2. Is het grootste natuurlijke getal in een n-bit voorstelling $2^n?1$?
3. Is 01111111 de one's complement voorstelling van -128 in een 8-bit voorstelling?
4. Zijn de getallen $(1010100011001)_2$ en $(12431)_8$ gelijk aan elkaar?
5. Is 218 het tussenresultaat als je -37 omzet naar one's complement in een 8-bit voorstelling?
6. Is het kleinste getal bij one's complement 2^{n-1}_1 in een n-bit voorstelling?
7. Als je weet dat een Floating Point voorstelling B=4, e=7, m=37 en het decimaal punt voor bit nummer 5 heeft, is de nauwkeurigheid dan $2?17$?
8. Stel dat je beschikt over een achttallig geheugen, waarvan de woordlengte 44 bits lang is. De exponent is 7 bits lang en staat in binary offset. Geef al de specificaties, tekening, absolute fout, normalizatie, nauwkeurigeid en het bereik als je weet dat het decimaal punt achteraan staat.
9. Hoe wordt -0.05 voorgesteld in het bovenvermelde geheugen?
10. Je hebt een Floating Point voorstelling met B = 2, e = 3, m = 3 en het decimaal punt vooraan. Liggen dan alle getallen tussen 2^3 en $2?5$?
11. Stelt 01110110 in binary offset met een 8-bit voorstelling het getal -10 voor?

Praktijk UNIX

1. Maak een script dat een lijst strings als parameters inleest en een aantal bewerkingen op deze lijst toepast naargelang de opties die aan het script worden meegegeven. De volgende lijst geeft een overzichtvan de opties en de bewerkingen die het script moet kunnen verwerken:

    - **-l** druk de lijst op het scherm
    - **-i** druk de lijst achterstevoren op het scherm

- **-a** druk de eerste en laatste string van de lijst op het scherm (indien maar 1 string wordt meegegeven aan het script, druk je deze twee keer af)
- **-m** de string in het midden van de lijst wordt afgedrukt (indien er een even aantal strings wordt meegegeven, druk je maar 1 string af)

Er is niet op voorhand bekend hoeveel strings er worden meegegeven. Het moet mogelijk zijn meerdere opties mee te geven. Een voorbeeld van het gebruik: myscript -l -mstr1str2str3. Dit moet het volgende als uitvoer geven:
str1str2str3
str2
(Hint : maak gebruik van arrays om de parameters in op te slaan)

2. Maak een script dat een rij getallen als parameter inleest en hiervan een aantal grootheden berekent naargelang de optie die het script meekrijgt. De volgende lijst geeft een overzicht van de opties en de grootheden die berekend moeten worden.
   - **-s** de som van de getallen
   - **-p** het product van de getallen
   - **-g** het grootste getal
   - **-k** het kleinste getal
   - **-m** het gemiddelde van de getallen

   Er is niet op voorhand bekend hoeveel getallen er worden meegegeven. Het moet mogelijk zijn meerdere opties mee te geven. Een voorbeeld van het gebruik:

   ```
   #> myscript -k -p 12 4 20 31 1 kleinste: 1 product: 29760
   ```

3. Maak een script dat een bestand inleest dat twee rijen getallen bevat. Het script moet deze twee rijen getallen van plaats verwisselen zonder een extra bestand te gebruiken. De naam van het bestand wordt als parameter aan het script meegegeven. Het bestand dat moet worden ingelezen ziet er als volgt uit:

   ```
   12 23 23 34 34 45 45 56 56 67 67 78 78 89 89 90
   ```

   Maak zelf een dergelijk bestand aan om je script te testen. Het script verwisselt de kolommen en drukt ze op het scherm op de volgende manier:

   ```
   23 12 34 23 45 34 56 45 67 56 78 67 89 78 90 89
   ```

   (Hint : maak gebruik van arrays en de for-lus om de rijen van plaats te verwisselen.)

Ontvangen van "http://www.winak.be/tuyaux/index.php?title=Computersystemen_en_-architectuur"
Categorieën: Informatica | 1BINF

Discrete Wiskunde

Uit Encyclopedia Academia

Bespreking

Discrete Wiskunde	
Professor	Guy Van Steen
Richting	Informatica
Jaar	1BINF

Het eerste wiskunde-vak van de opleiding Informatica, maar wees gerust, in dit vak sparen ze je nog een beetje. Dit vak wordt gedoceerd door professor Van Steen en zoals je zal merken of al hebt gemerkt: wordt er in de lessen als een sneltrein op het bord geschreven en zal je elke les alles zo snel mogelijk moeten overschrijven van wat er op bord staat. Het leermateriaal bestaat dan ook uit uw eigen nota's en uit een handboek. Dit handboek is best goed, de leerstof wordt er goed en correct in uitgelegd, er staan veel voorbeelden in. Soms wijdt men in dit boek wat teveel uit over bepaalde zaken die je misschien eens kunt lezen maar voor de rest niet te veel van moet aantrekken.

Op het theorie-examen wordt er naar de goede gewoonten van professor Van Steen praktisch enkel om definities, stellingen en bewijzen gevraagd. Het is dus best eenvoudig om een samenvatting te maken met daarin al deze informatie en dit te leren. Luister ook steeds naar de tips die professor Van Steen geeft, heel vaak geeft hij aan bij bepaalde stukken uit de cursus dat dat deel belangrijk is. Als hij dit zegt, noteer dit dan ook want dit zijn de delen waaruit hij vragen stelt.

De praktijk volgt gewoon de theorie van het boek en als je de voorbeelden uit het boek begrijpt zullen deze oefeningen ook niet echt een struikelblok vormen. Let wel op om bij de theorie en praktijk zo formeel mogelijk te antwoorden en alle uitzonderingen te noteren. Dus als je antwoord $\frac{1}{x}$ is (het is maar een voorbeeld), noteer er dan bij dat dit geldt $\forall x \neq 0$. Ook het gebruik van symbolen in plaats van ellenlange zinnen wordt aangemoedigd.

Puntenverdeling

Theorie: 10/20. Praktijk 10/20.

Examenvragen

Academiejaar 2010 - 2011 - 2de zittijd

Theorie Examen

1. Gebruik voor volgende vragen de exacte wiskundige notatie (geen pijlen, venndiagrammen, ...)
    1. Verklaar de termen injectief, surjectief, bijectief
    2. Geef een surjectieve functie die niet injectief is en leg uit
    3. Geef een injectieve functie die niet surjectief is en leg uit
    4. Geef een relatie die geen functie is en leg uit
2. 1. Definieer het zwak inductieprincipe

    2. Bewijs via dit principe een formule voor de som van de hoeken in een convexe n-hoek
3. Geef de formule voor het aantal mogelijkheden om b identieke ballen in n verschillende dozen te stoppen en bewijs deze
4. 1. Definieer de binomiaaldistributie
    2. Definieer een random variabele op de kansruimte zodat de bijhorende kansdistributie de negatieve binomiaaldistributie is en leg uit
5. 1. Geef de definities van E(X) en Var(X)
    2. Geef de verwachtingswaarden van de binomiaal en de negatieve binomiaaldistributies (ZONDER bewijs)

Academiejaar 2010 - 2011 - 1ste zittijd

Theorie Examen

1. 1. Formuleer de voorwaarden waaraan een relatie van een verzameling A naar een verzameling B moet voldoen om een functie te zijn. Gebruik de correcte wiskundige notaties en terminologie, dus geen Vendiagrammen, pijlen, etc ...
    2. Geef een voorbeeld van een relatie tussen oneindige verzamelingen die geen functie is. Leg uit!
    3. Geef een voorbeeld van een injectieve functie tussen de verzamelingen die je gebruikte in de vorige opgave. Toon aan waarom de functie injectief is.
2. Bewijs met volledige inductie de volgende ongelijkheid voor alle $n\in\mathbb{N}$ en voor alle $x\in\mathbb{R}_{\geq 0}$: $(1+x)^n \geq 1+nx$. Waar gebruik je de voorwaarde $x\geq 0$?
3. Geef een combinatorisch bewijs voor de volgende gelijkheid: $[\binom?]nk=[\binom?]n-1k+[\binom?]n-1k-1$. Hierbij zijn n en k natuurlijke getallen met $k<n$.
4. Op hoeveel verschillende manieren kan men b identieke ballen verdelen over u onderscheidbare dozen? Bewijs!
5. 1. Definieer het begrip random variabele en de bijhorende kansdistributiefunctie.
    2. Zij $U\subset\mathbb{R}$ een eindige verzameling en zij $f:U\Rightarrow[0,1]$ een functie. Aan welke voorwaarden moet f voldoen om distributiefunctie van een random variabele te zijn? Leg uit!
    3. Een telefooncentrale krijgt gemiddeld λ gesprekken per minuut te verwerken. Wat is de kans dat de centrale in de komende minuut k gesprekken te verwerken krijgt. Bewijs uw antwoord!
6. 1. Definieer het begrip voorwaardelijke kans.
    2. In een doos zitten 2 witte en 2 zwarte (onderscheidbare) ballen. Blindelings worden twee ballen uit de doos gehaald. Wat is de kans dat eerst gekozen bal zwart is? Wat is de kans dat de eerst gekozen bal zwart is als je weet dat de tweede bal ook zwart is? Leg uit! Beschrijf duidelijk de kansruimte en de gebeurtenissen die je gebruikt. (zonder teruglegging)

Praktijk Examen

1. Geef aan welke van de volgende relaties functies zijn, en indien ja, of ze injectief, surjectief, bijectief zijn. Noem hierbij $2\mathbb{Z}$ de verzameling van even gehele getallen. Bewijs je beweringen.
    1. $f:\{0,1,2,3,...,63\}\rightarrow\{0,1\}^n:n\rightarrow(a_0,a_1,a_2,a_3,a_4,a_5)$ waarbij $(a_5a_4a_3a_2a_1a_0)_2$ de binaire voorstelling is van het getal n.
    2. $g:\mathbb{N}\times\mathbb{N}\rightarrow\mathbb{N}:(n,m)\rightarrow m2^n$.
    3. $h:(\mathbb{N}\setminus\{0\})\times(\mathbb{N}\setminus 2\mathbb{Z})\rightarrow\mathbb{N}:(n,m)\rightarrow m2^n$.

2. De uitdrukking $1*2+2*3+3*4+4*5+...+n(n+1)$ kan je schrijven in een gesloten formule. Bepaal deze formule via de methode van onbepaalde coëfficiënten. Bewijs de formule ook via inductie.
3. Drie verzamelingen A, B en C voldoen aan de volgende eigenschappen:
   - $|A|=100$, $|B|=50$ en $|C|=48$.
   - Het aantal elementen dat slechts tot één van de drie verzamelingen behoort is het dubbel van het aantal elementen dat tot precies twee van de drie verzamelingen behoort.
   - Het aantal elementen dat slechts tot één van de drie verzamelingen behoort is het drievoud van het aantal elementen dat tot elk van de drie verzamelingen behoort.
4. Bij het spel Yahtzee gooit men met 5 zeszijdige dobbelstenen. We gebruiken de volgende terminologie:
   - Een full house is een worp waarbij 3 dobbelstenen een gelijk aantal ogen hebben en de twee andere dobbelstenen ook een gelijk aantal ogen hebben.
   - Een carré is een worp waarbij er minstens 4 dobbelstenen hetzelfde aantal ogen hebben.
   - Een yahtzee is een worp waarbij de 5 dobbelstenen hetzelfde aantal ogen hebben. Bepaal de kans dat men bij één worp meteen een full house, een carré of een yahtzee gooit?
5. Een HDTV bestaat uit 103 componenten, die elk met een kans van $0{,}03\%$ defect zijn. De HDTV werkt slechts als al zijn componenten werken. Wat is de kans dat de HDTV defect is?
6. Louis neemt regelmatig het vliegtuig en hij houdt ervan zijn zitplaats te upgraden naar eerste klasse. Hij heeft gemerkt dat als hij minstens 2 uur op voorhand inchecked, dat hij dan 75% kans heeft op een upgrade. In het andere geval is de kans op een upgrade slechts 35%. Met Louis zijn zeer druk schema kan hij slechts bij 40% van zijn vluchten meer dan 2 uur op voorhand inchecken. Veronderstel nu dat Louis geen upgrade heeft kunnen hebben tijdens zijn laatste vlucht. Wat is dan de kans dat hij minstens 2 uur voor het vertrek incheckte?
7. De bits x, y, z en w zijn de bits die als input gebruikt worden voor een logische circuit. Samen stellen ze het binair getal $xyzw$ voor. Veronderstel dat het circuit 0 als output geeft indien $xyzw$ het kwadraat is van een geheel getal en 1 als output geeft indien niet.
   1. Gebruik Karnaugh maps om een zo kort mogelijke Boolse uitdrukking te krijgen voor deze output.
   2. Teken het bijbehorende logische circuit.

Tussentijdse Test

Theorie

1. 1. Definieer het begrip equivalentierelatie. Hoe definieert men de bijbehorende quotientverzameling?
   2. Geef een voorbeeld van een equivalentierelatie op $\mathbb{Z}$ die een eindige quotientverzameling geeft. Beschrijf de quotientverzameling zo eenvoudig mogelijk. Hoeveel elementen telt deze quotientverzameling? Leg telkens uit waarom!
2. 1. Formuleer het sterk inductieprincipe.
   2. Toon aan met volledige inductie dat elk natuurlijk getal dat groter of gelijk is aan 8 een som is van een 3-voud en een 5-voud. Laat duidelijk zien hoe je het inductieprincipe toepast.

Praktijk

1. Toon aan dat voor $n \geq 1$ steeds geldt dat 9 een deler is van $4^n+15n-1$.
2. Elk van de volgende grafieken komt overeen met een functie van $\mathbb{R}$ naar $\mathbb{R}$. Geef telkens een

functievoorschrift dat overeenkomt met deze grafiek.

Academiejaar 2009 - 2010 - 1ste zittijd

Theorie Examen

1. 1. Geef een voorbeeld van een equivalentierelatie op $\mathbb{Z}$. Toon aan dat dit wel degelijk een equivalentierelatie is. Is de quotiëntverzameling eindig of oneindig?
   2. Geef een voorbeeld van een oneindige verzameling met een partiële orderelatie die geen totale ordening is. Leg uit!
2. Bewijs de volgende formule met volledige inductie: $(1+x)^n \geq 1+nx$ voor alle $n \in \mathbb{N}$ en $x \in \mathbb{R}$ met $x \geq 0$. Geef duidelijk aan waar je gebruikt dat $x \geq 0$.
3. Geef zowel een algebraïsch als een combinatorisch bewijs van de volgende formule: $\binom{n-1}{k-1}+\binom{n-1}{k}=\binom{n}{k}$. Hierbij is $n \in \mathbb{N}_0$ en $1 \leq k \leq n-1$.
4. 1. Definieer het begrip *voorwaardelijke kans*.
   2. Een doos bevat twee witte ballen en twee zwarte ballen. Twee ballen worden uit de doos weggenomen. Wat is de kans dat de eerste weggenomen bal wit is? Verklaar. Wat is de kans dat de eerst weggenomen bal wit is, gegeven dat de tweede bal ook wit is? Verklaar. Geef een duidelijke beschrijving van de kansruimte, de kansmaat en de gebeurtenissen die je hier gebruikt.
5. 1. Definieer het begrip *random variabele op een kansruimte*. Hoe definieert men de kansdistributiefunctie van een random variabele?
   2. Geef een voorbeeld van een kansruimte met een random variabele zodat de bijbehorende kansdistributiefunctie de binomiaaldistributie is. Leg uit!

Praktijk Examen

1. Veronderstel dat P de verzameling van proffen is van deze universiteit en C de verzameling van alle cursussen die aan deze universiteit gedoceerd worden. De relatie $R \subset C \times P$ wordt beschreven door “... wordt gedoceerd door ...”.

   1. Wat wil het zeggen dat deze relatie een functie is?

   2. Indien R overeenkomt met een functie, wat wil het dan zeggen dat deze functie injectief is?

   3. Indien R overeenkomt met een functie, wat wil het dan zeggen dat deze functie surjectief is?

   4. Kan deze relatie een equivalentierelatie of een partiële ordening zijn?

2. Bewijs *via inductie* dat n^2-1 deelbaar is door 8 voor elk oneven getal *n* met $n>0$.

3. Een book van 500 pagina’s bevat 500 drukfouten. Gebruik de Poissonverdeling om de kans te bepalen dat er op een willekeurige pagina drie of meer drukfouten staan.

4. Een wandelaar kan uit *n* wegen kiezen om van A naar B te wandelen. De wegen zijn genummerd 1 tot *n*. Indien de wandelaar weg i neemt, splitst de weg onderweg in $2i$ wegen, waarvan er slechts 1 tot B leidt. Als de wandelaar zowel in het begin als in het midden willekeurig een weg kiest. Indien gegeven is dat de wandelaar aankomt, wat is dan de kans dat hij pad i heeft gekozen.

5. Je beschikt over 6 (6-zijdige) dobbelstenen met de waarden 1, 1, 2, 2, 3, 3 op. Als je met

deze 6 dobbelstenen tegelijk gooit, wat is de kans dat elk van de 3 cijfers minstens 1 keer voorkomen?

6. Gegeven de volgende logische tabel

p	q	r	s	**p**
1	1	1	1	1
1	1	1	0	1
1	1	0	1	0
1	1	0	0	1
1	0	1	1	1
1	0	1	0	1
1	0	0	1	0
1	0	0	0	0
0	1	1	1	1
0	1	1	0	1
0	1	0	1	1
0	1	0	0	1
0	0	1	1	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	0

    1. Gebruik Karnaugh Maps om een zo kort mogelijke Boolse uitdrukking in DNV te geven die overeenkomt met deze logische tabel.

    2. Teken het bijbehorende logische circuit.

Academiejaar 2009 - 2010 - 2de zittijd

Theorie Examen

1.
    1. Geef de definitie van injectief, surjectief en bijectief.
    2. Geef een voorbeeld van een functie $f:\mathbb{Z}\to\mathbb{Z}$ die injectief is, maar niet surjectief.
    3. Geef een voorbeeld van een functie $f:\mathbb{Z}\to\mathbb{Z}$ die niet injectief is, maar wel surjectief.
2. Bewijs via inductie: de som van de hoeken van een convexe n-hoek.
3. Geef het combinatorisch bewijs van $n*2^{(n-1)}=\sum_{k=0}^{n} k\binom{n}{k}$.
4. Formuleer de stelling van Bayes en geef een uitgewerkte toepassing.
5.
    1. Geef de definitie van een binomiaaldistributie. Geef het definitiegebied, de beeldverzameling en leg uit hoe deze gebruikt wordt.
    2. Geef een kansruimte, definieer hierop een random variabele zodat de distributie de binomiaaldistributie is en leg uit.

Academiejaar 2008 - 2009 - 1ste zittijd

Theorie Examen

1. Een vraag over Functies (een aantal definities met minimum, maximum, partieel geordende

verzameling)
2. Bewijs $2^n=\sum_{k=0}^{n}C(n,k)$ met inductie
3. Bewijs $\sum_{k=0}^{n}(kC(n,k))=n2^{(n-1)}$ combinatorisch
4. Geef de formule voor de alternatieve binomiaal coëfficient $C(-n,k)$ voor $n>0$ in functie van de gewone binomiaal + bewijs
5. Geef de definitie van
    1. Een random variabele of een kansruimte
    2. De kansdistributie die wordt geassocieerd met een random variabele
    3. De variantie
6.
    1. Geef de definitie van Bn,p
    2. Geef een voorbeeld van een kansruimte waarvan de random veriabele wordt geassocieerd met de negatieve binomiaal coëfficient

Praktijk Examen

1. Hoeveel getallen $<$ 1 miljoen zijn niet deelbaar door 2, 3 of 5.

2. Een politie-agent moet de baas van een gangsterbende volgen wanneer ze hun vergaderplek verlaten (een huis). Hij weet alleen dat de gangsters met 5 zijn, dat ze allemaal een verschillende lengte hebben en dat de baas de grootste is. Om veiligheidsredenen komen de gangster apart - om het kwartier - buiten. De agent beslist om de eerste 2 te laten gaan en dan de eerste te volgen die groter is. Hoeveel kans heeft hij om de baas te volgen?

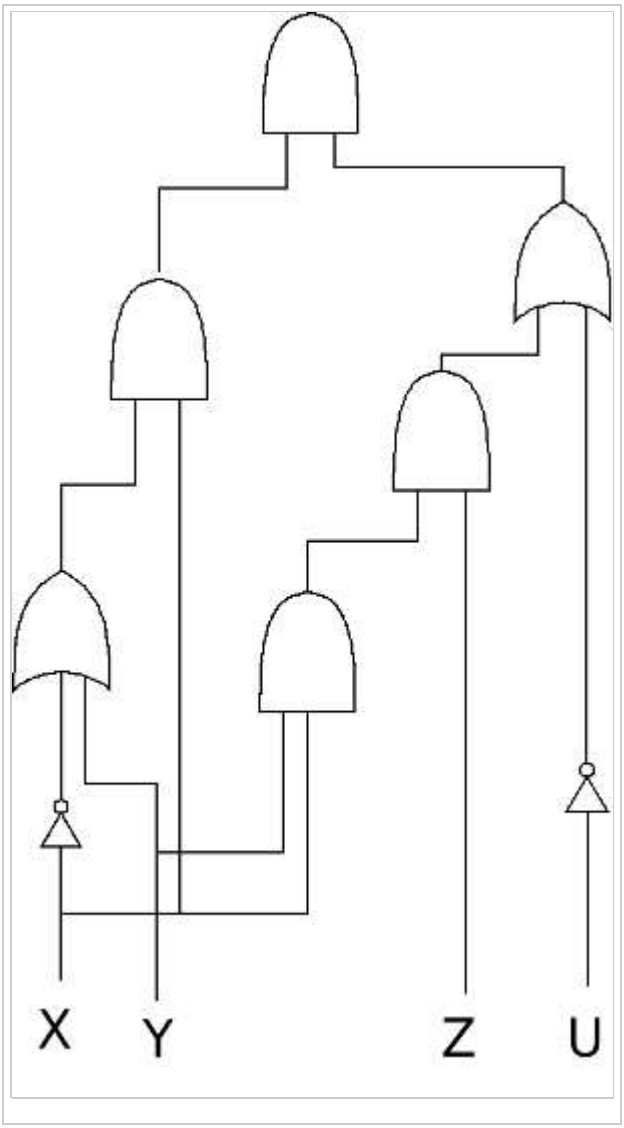
3. Je hebt een eerlijke (zeszijdige) dobbelsteen. Je hebt 3 Urnes.
   Urne I (4 zwarte en 3 witte ballen)
   Urne II (2 Zwarte en 5 witte)
   Urne III (7 zwarte)

   Als je 1 gooit pak je een bal uit I. Als je 2 of 3 gooit uit II en 4, 5, 6 uit III.

    1. Als je een zwarte bal hebt gepakt, wat is dan de kans dat je een 6 hebt gegooid.

    2. Bereken de variantie van de dobbelsteen als je een zwarte bal hebt gepakt.

4. Gegeven het volgende logische circuit (Figuur fig:circuit)



1. Geef een booleanse zin die hiermee overeenkomt.
2. Vereenvoudig deze met Karnaugh-maps en teken het versimpelde circuit.

Tussentijdse Test

Theorie Tussentijdse Test

1.
   - Geef de definities voor de begrippen injectieve functie en totale ordening
   - Toon aan dat de samenstelling van 2 injectieve functies weer injectief is
   - Geef een voorbeeld van een partieel geordende verzameling die niet totaal geordend is. Leg uit!
   - Hoe ziet het hassediagram van een eindige totaal geordende verzameling eruit.
2. Zij x element van $\mathbb{R}$ een positief getal. toon aan met volledige inductie dat $(1+x)^n \geq 1+nx$ (laat duidelijk zien waar je gebruikt dat $x \geq 0$)

Praktijk Tussentijdse Test

1. De rij $A_0, A_1, A_2, \ldots$ wordt gegeven door de voglende recursieve definitie: $A_0 = 1, A_n = A_{n-1} + 2n^2 - 4n + 3$

- Schrijft de termen $A_0,...A_5$ van deze rij uit en bepaal een veeltermvoorschrift voor deze rij via de methode van de onbepaalde coëfficiënten.
- Bewijs via inductie dat dit voorschrift inderdaad het voorschrift is van deze rij.

2. Los op in $\mathbb{Z}$: $6x+3\equiv 1 (mod 8)$

Academiejaar 2007 - 2008 - 1ste zittijd

Theorie

1. ## Definieer de begrippen orderelatie en equivalentierelatie op een verzameling A.
    1. Geef een concreet voorbeeld van een equivalentierelatie op verzameling [\textbf?]$\mathbb{Z}$.
    2. Geef een concreet voorbeeld van een orderelatie op een eindige verzameling die geen totale ordening is.
    3. Is er een maximum in dit voorbeeld? Zo ja, duid dit aan.
    4. Zijn er minimale elementen in dit voorbeeld? Zo ja, duid ze aan.
2. Bewijs volgende formule met volledige inductie: $\sum_{k=0}^{n}\binom{n}{k}=2^n$.
3. Gegeven zijn verschillende objecten O_1, …, O_t. We kiezen d_1 keer het object O_1, d_2 keer het object O_2, … en d_t keer het object O_t. Geef en bewijs een formule die het aantal mogelijke ordeningen geeft van de gekozen objecten.
4. Geef een exacte formulering van de stelling van Bayes. Bewijs de stelling!
5. 1. Geef de definitie van de distributiefunctie van een random variabele X op een eindige kansruimte S.
    2. Geef de definities van de verwachtingswaarde van een randomvariabele en van de bijbehorende distributiefunctie op een eindige kansruimte.
    3. Geef de definitie van de binomiaaldistributie Bn,p (met $0\leq p\leq 1$). Wat is de verwachtingswaarde van deze distributie? (Geen bewijs!)

Praktijk

Stijn Symens als Assistent

1. Geef van de volgende relaties aan of het equivalentierelaties zijn, partieel geordende verzamelingen zijn, beide zijn of geen van beide zijn. Geef in het geval van equivalentierelatie de bijbehorende partities , in het geval van partiële ordening het bijbehorende hasse diagram.

   Zie figuur zoals in vb 1.17 pg 6 van de oefeningen.

2. In een bepaalde richting kan je als student een aantal keuzevakken kiezen. Drie van die keuzevakken zijn talen. In een klas van 100 studenten zijn er 35 die frans volgen, 42 die Spaans volgen, 43 die duits volgen, 17 die frans en spaans volgen, 13 die frans en duits volgen, 15 die spaans en duits volgen en 20 die geen taal hebben gekozen.

    1. Hoeveel studenten volgen frans of duits , maar geen spaans.

    2. Hoeveel studenten volgen precies 1 taal?

    3. Hoeveel studenten volgen precies 2 talen?

3. Bewijs met inductie dat 13 een deler is van $8^{2n}-5^{2n}$ voor elke n element van $N\ 0$.

4. Een multiple-choice examen (op 50 punten) heeft 10 vragen, 3 met 4 mogelijke antwoorden en 3 met 5 mogelijke antwoorden en 4 met 6 mogelijke antwoorden. Een student krijgt

steeds 5 punten per correct antwoord, 0 per vraag die hij openlaat en -1 bij een foutief antwoord. Een student gokt op elke vraag. Wat is zijn verwacht aantal punten?

5. Een 2- Euromuntstuk wordt opgegooid. Als het kop is, wordt een paar dobbelstenen geworpen en de speler krijgt het aantal euro dat overeenkomt met het gezamelijk aantal ogen van de worp. Als het munt is, worden 3 munten opgeworpen en de speler krijgt 4 euro per kop die gegooid wordt. Als de speler 8 euro heeft gewonnen, wat is dan de kans dat het oorspronkelijk 2-euromuntstuk op kop landde?

6. Gegeven de volgende logische tabel. (Het was een tabel met 4 letters, p, q, r ,s)

    1. Teken Karnaugh Maps om een zo kort mogelijke Boolse uitdrukking te geven die overeenkomt met deze logische tabel.

    2. Teken het bijbehorende logische circuit.

Academiejaar 2007 - 2008 - 2de zittijd

Theorie

1. Geef de stelling van Bayes en illustreer deze met een voorbeeld (dus een toepassing geven en uitwerken).
2. Geef de definitie van een transitieve relatie.

Praktijk

1. Gegeven de verzameling $A=\{a,b,c,d,e,f\}$.

    1. Hoeveel injectieve functies zijn er van A naar A?

    2. Hoeveel relaties zijn er op A?

    3. Hoeveel relaties zijn er op A die symmetrisch en reflexief zijn?

2. In een paardenkoers zijn er 4 deelnemende paarden. Op goeveel verschillende manieren kunnen die over de streepo komen? Het is hierbij mogelijk dat verschillen,de paarden exact op hetzelfde tijdstip over de streep komen (en er dus bv. een gedeelde tweede plaats is).

3. Het aantal verkeersongevallen in de gemeente “Accidorp” werd in 2007 geleten. Er werden statistieken bijgehouden van het aantal ongevallen per dag, en dit is weergegeven in de volgende tabel:

Aantal verkeersongevallen x	Aantal dagen met x verkeersongevallen
0	206
1	108
2	41
3	10
4 of meer	0
totaal	356

Gebruik een gekende verdeling, die typisch is voor dit soort gegevens, om deze gegevens te benaderen. Wat zou, volgens die verdelingsfucntie, de kans zijn dat er 4 of meer

verkeersongevallen op een dag gebeuren.

4. Bewijs via inductie dat: $1+\frac{1}{4}+\ldots+\frac{1}{n^2}<2-\frac{1}{n}$

5. Veronderstel dat we 2 zakken hebben met witte en zwarte ballen in. In de ene zak zitten 3 keer meer witte ballen dan zwarte. In de andere zak zitten 3 keer meer zwarte ballen dan witte. Veronderstel dat we willekeurig een zak kiezenn en dan uit die zak willekeurig 5 ballen nemen (waarbij we een bal steeds terugsteken als we hmm genomen hebben). Het resultaat is dat we 4 witte en 1 zwarte bal getrokken hebben. Wat is de kans dat de ballen uit de zak met vooral witte ballen in getrokken zijn?

6. Gegeven de volgende logische tabel

p	q	r	s	p
1	1	1	1	0
1	1	1	0	1
1	1	0	1	0
1	1	0	0	0
1	0	1	1	0
1	0	1	0	1
1	0	0	1	0
1	0	0	0	0
0	1	1	1	0
0	1	1	0	1
0	1	0	1	0
0	1	0	0	0
0	0	1	1	0
0	0	1	0	1
0	0	0	1	1
0	0	0	0	1

    1. Gebruik Karnaugh maps om een zo kort mogelijke Boàolse uitdrukking te geven die overeenkomt met deze logische tabel.

    2. Teken het bijbehorende logische circuit.

Modelvraag

Theorie

1. Formuleer het *Principe van Inclusie en Exclusie* voor 2 verzamelingen A en B.
   Geef een veralgemening van dit principe voor meerdere verzamelingen $A_1,[\dots?],A_t$.
2. In de kanstheorie bestaat een regel die veel gelijkenis vertoont met het principe van inclusie en exclusie. Geef en bewijs deze regel voor twee gebeurtenissen A en B in een kansruimte (S,Pr).
3. Definieer het begrip *voorwaardelijke kans*.
   Als A en B disjuncte gebeurtenissen zijn met een kans die niet 0 is, kunnen deze gebeurtenissen dan onafhankelijk zijn? Waarom?

4. Wat is de verwachtingswaarde van de negatieve biniomiaaldistributie met orde 1?
   ($f(k)=p.q^{k-1}$, $k\geq 1$)
   Bewijs uw antwoord.

Academiejaar 2004 - 2005 - 1ste zittijd

Praktijk

1. Bewijs per inductie dat $\sum_{S\subset[n]}2^{|S|}=3^n$ Waaraan is dan $\sum_{S\subset 2[n]}2^{|S|}$ gelijk?
2. Bewijs dat volgende recursief gedefineerde formule $\begin{matrix}V(1)=0\\V(2)=1\\\forall k\geq 3: V(k)=(k-1).(V(k-1)+V(k-2))\end{matrix}$ gelijk is aan volgende som. $V(k)=k!\sum_{i=2}^{k}\frac{(-1)^i}{i!}$
3. #* Hoeveel nieuwe woorden bekomen we door de letters van het woord {parallellogram} te herschikken?
   - Hoeveel van deze woorden zullen beginnen met een *a*?
   - Hoeveel van deze woorden zullen beginnen en eindigen met een klinker?
   - In hoeveel van deze woorden zullen er geen opeenvolgende *l*'s of *r*'s staan?
4. Hoeveel mogelijke oplossingen zijn er voor volgende som. $x+y+z+u=15$
   - Als $x,y\in\mathbb{N}$ en $z,u\in\mathbb{N}_0$.
   - Als $x,y\in\mathbb{N}$ en $z,u\in\mathbb{N}_0$ en $x+y=7$ of $x+y=8$.
   - Als $x\in\mathbb{N},y\in\mathbb{N}_0$ en $z,u\in\mathbb{Z}$ met $z\geq -1$ en $u\geq -2$.
5. Stel dat elke persoon twee genen bezit die de haarkleur bepalen.
   Er zijn twee verschillende soorten, een gen voor blond en een gen voor zwart haar, die beiden evenveel voorkomen.
   We weten dat het blonde gen recessief en het zwarte gen dominant is, dit wil zeggen dat als een persoon één gen voor blond en één voor zwart haar bezit, dat deze persoon dan zwart haar zal ontwikkelen.
   Uiteraard weten we ook dat een kind één gen van elke ouder zal krijgen.
   Beantwoord nu de volgende vragen.
   - Wat is de kans dat een persoon blond haar heeft?
   - Wat is de kans dat een persoon zwart haar heeft indien zijn/haar vader ook zwart haar heeft?
   - Wat is de kans dat een persoon zwart haar heeft indien zijn/haar vader een andere haarkleur heeft dan zijn/haar moeder?
   - Wat is de kans dat beide ouders zwart haar hebben indien hun eerste kind blond is?
   - Stel een gezin met 6 kinderen waarbij de vader en moeder verschillende haarkleur hebben. Wat is de kans dat dit gezin twee maal zoveel blond als zwartharige kinderen heeft?
   - Wat is de kans dat in dit gezin al de blondharigen meisjes zijn en al de zwartharigen jongens?
   - Stel een koppel waarvan vader en moeder verschillende haarkleur hebben wat is de kans dat er pas na vier andere kinderen het tweede blonde kindje geboren wordt?
6. Een bedrijf wil een wedstrijd organiseren waarbij er verschillende geldprijzen te winnen zijn. Elke deelnemer koopt een lotje van 0,50 euro, en maakt kans op volgende prijzen.
   De hoofdprijs bedraagt 1500 euro en dit wordt toegekend aan een lot getrokken uit alle gekochte loten. Op dezelfde manier worden er 2 prijzen van 500 euro en 25 prijzen van 100 euro geloot.
   Er wordt geloot met teruglegging.
   - Stel dat het bedrijf minimaal 1000 euro aan deze loterij wil verdienen, hoeveel loten moeten er dan verkocht worden? Wat is de verwachte netto winst per deelnemer indien er 20.000 loten worden verkocht, bereken ook de variantie.

- Stel nu dat bovenop de te winnen prijzen, elke honderdste koper al 10 euro wint. Hoeveel loten moet het bedrijf dan minimaal verkopen? Stel dat er weer 20.000 loten zullen worden verkocht, zal de verwachte nettowinst per deelnemer hier lager of hoger liggen, wat met de variantie?

Academiejaar 2004 - 2005 - 2de zittijd

Theorie

1. - [(a)] Definieer de begrippen *injectieve functie*, *surjectieve functie* en *bijectieve functie*.
   - [(b)] Geef een voorbeeld van een functie die wel surjectief maar niet injectief is.
   - [(c)] Geef een voorbeeld van een functie die wel injectief maar niet surjectief is.
   - [(d)] Geef een voorbeeld van een relatie die geen functie is.

   Gebruik bij de formulering alleen de standaard notatiesystemen voor verzamelingen en functies, dus *geen* Venndiagrammen.
2. Geef een combinatorisch bewijs voor de volgende formule: $[\binom?]nk=[\binom?]n-1k+[\binom?]n-1k-1$ waarbij $n,k\in\mathbb{N}$ en $k\leq n-1$.
3. Op hoeveel manieren kan men b identieke ballen verdelen over n verschillende (en onderscheidbare) dozen. Bewijs!
4. - [(a)] Formuleer de stelling van Bayes. (Geef de volledige formulering, dus niet alleen een formule.)
   - [(b)] Geef een voorbeeld van een toepasing van deze stelling.
5. - [(a)] Definieer de begrippen *verwachtingswaarde* en *variantie* van een random variabele op een eindige kansruimte.
   - [(b)] Bereken de verwachtingswaarde van de binomiaaldistributie.

Ontvangen van "http://www.winak.be/tuyaux/index.php?title=Discrete_Wiskunde"
Categorieën: Informatica | 1BINF

Gegevensstructuren

Uit Encyclopedia Academia

Bespreking

Gegevensstructuren	
Professor	Els Laenens
Richting	Informatica
Jaar	1BINF

Theorie

De theorie van dit vak is eigenlijk puur bedoeld om de praktijk tot een goed einde te brengen, dit vak wordt gedoceerd door Els Laenens en alhoewel soms wat traag, legt ze alles goed en duidelijk uit. Het leermateriaal bestaat uit een verzameling van slides en aangeraden is om het handboek te kopen. Dit handboek is niet echt nodig en zul je ook niet zo vaak gebruiken, maar kan wel nuttig zijn aangezien de meeste oplossingen van de praktijkopgaves hier letterlijk instaan (in C++ weliswaar). Het theorie-examen is niet echt moeilijk, enkele technieken een beetje vergelijken met elkaar.

Praktijk

Hier gaat het vak eigenlijk om. Buiten de oefeningensessies die je helpen bij het oplossen van de programmeeropdrachten, krijg je 5 opdrachten waarin je een bepaalde gegevensstructuur moet implementeren. Deze opdrachten zijn niet zo moeilijk maar het is wel uitermate belangrijk om ze allemaal op tijd in te zenden. Als je een opdracht vergeet in te zenden of deze niet hebt afgewerkt en ze daarom maar niet inzendt, dan ben je al vast en zeker gebuisd. Dus zelfs al werkt alles niet compleet, stuur toch je opdracht in als je wilt slagen voor dit vak. Na het hebben ingestuurd van je oplossing krijg je een andere student zijn oplossing die je dan moet testen en beoordelen. Vervolgens krijg je de beoordeling van een andere student op jouw oplossing toegestuurd die je dan op zijn beurt moet beoordelen of verantwoorden.

Puntenverdeling

Praktijk: 14/20 (waarvan 10/20 op de oplossingen, 3/20 op de beoordelingen en 1/20 op de verantwoording). Theorie: 6/20.

Examenvragen

Academiejaar 2010 - 2011 - 1ste zittijd

Theorie

1. Bij de implementatie van het type hashtabel kies je best een tablesize die priem is. Waarom?
2. Stel een tabel op van alle geziene (zoek)algoritmen. Vermeld hierin hun worst case en average case efficiëncy.
    1. Bespreek de verschillende graden van efficiëntie intuitief.
    2. Schrijf pseudocode voor één van de algoritmes die je boven hebt vermeld en beschrijf hoe je aan de gekomen efficiëntie komt.
3. Wat is het aantal knopen voor een volle binaire boom op hoogte h?
    1. Wat is het totaal aantal knopen in een binaire boom van hoogte h?
    2. Bewijs je bevindingen.

4. Geef de formele definitie van:
    1. Een algemene zoekboom
    2. Een B-boom van graad m
    3. Wanneer is m optimaal?
    4. Schrijf pseudocode voor een tableretrieve.

Praktijk

1. Afbeelding Missing | Geef steeds de structuur van de boom weer.
    1. Voeg toe (in deze volgorde): 75,20,16,48,1.
    2. Delete: 70.

1. Afbeelding Missing | Gegeven de volgende graaf:
    1. Doorloop de boom BFS en geef de bezochte volgorde.
    2. Geef de BFS spanning tree.
    3. Geef de minimum spanning tree.

1. Zet volgende array om in een heap d.m.v. heaprebuild.
    1. 14 20 3 16 24 9 12 8 18

1. Is dit een correcte Red-Black tree? Afbeelding Missing:
    1. Indien ja, voeg 18 toe aan de boom en geef de resulterende boom weer.
    2. Indien nee, leg uit waarom.

Labotoets

1. Vertrek van je reeds geschreven ADT binaire zoekboom. De unieke zoeksleutels zijn die van het type INTEGER. Voeg er de volgende operaties aan toe:
    1. Geef het item met de grootste zoeksleutel weer.
    2. Geef het item met de kleinste zoeksleutel weer.
    3. Geef de gemiddelde waarde van de zoeksleutels.
2. Voer (als test) volgende reeks operaties uit:
    1. Creëer een lege binaire zoekboom.
    2. Voeg achtereenvolgens 12,18,14,6,4,9,20,17,5,7,15 toe.
    3. Bepaal de hoogte van de boom.
    4. Geef de boom weer, inorder.
    5. Geef het item met de kleinste sleutel.
    6. Geef het item met de grootste sleutel.
    7. Geef de gemiddelde waarde van de sleutels.
    8. Voeg achtereenvolgens 2,3,19 toe aan de boom en verwijder 7,20,6,12. Herhaal stap c -> g.
3. Implementeer het ADT minheap, gebruik makende van het ADT binaire zoekboom. De elementen opgeslagen in de heap zijn uniek en hebben een zoeksleutel van het type INTEGER. We hebben de volgende operaties nodig.
    1. Create
    2. Wis een minheap
    3. Bepaal of een minheap leeg is.
    4. Voeg een nieuw element toe aan de minheap.
    5. Verwijder het element met de kleinste zoeksleutel.
4. Voer de volgende testoperaties uit.
    1. Creëer een lege minheap.
    2. Voeg items met zoeksleutels 12,6,16,21,14,2,18,20,17 toe.
    3. Verwijder 3 keer achter elkaar het item met de kleinste zoeksleutel.

5. Pas het ADT minheap aan, zodat de zoeksleutels niet uniek hoeven te zijn. In dit geval wordt bij het verwijderen van een item het item verwijderd dat zich het langst in de heap bevindt. Schrijf hier zelf een testprogramma voor.

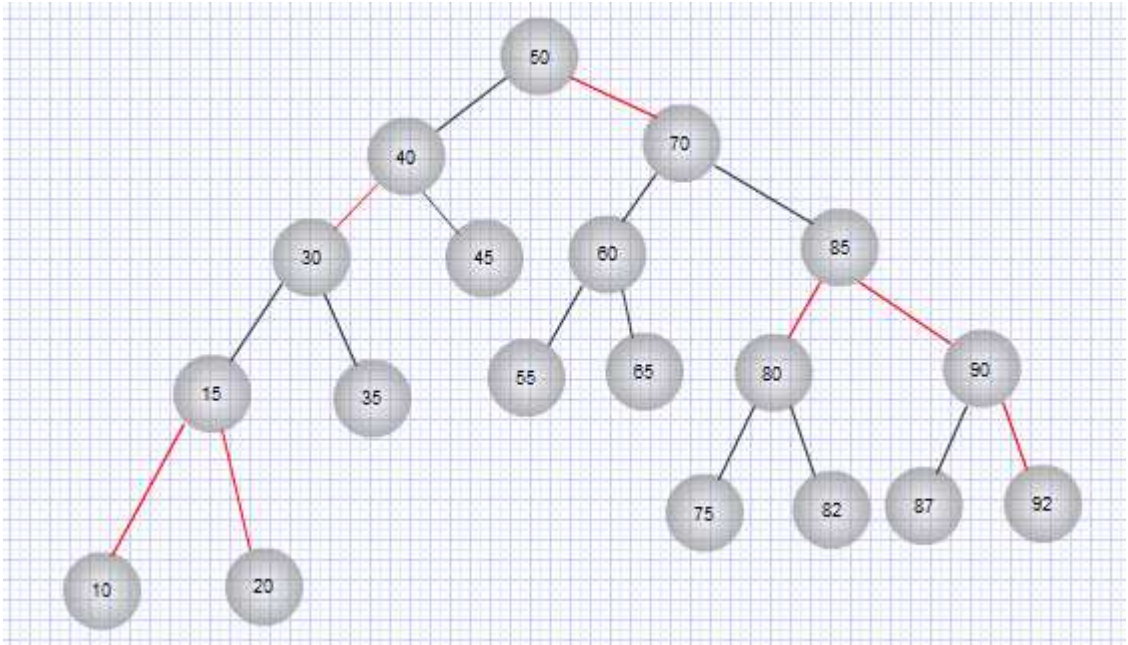
Academiejaar 2009 - 2010 - 1ste zittijd

Theorie

1. Bij double hashing geldt dat elke zoekreeks alle locaties in de hashtabel bezoekt als de tabelgrootte en stapgrootte relatief priem zijn, geef een concreet voorbeeld dat aantoont dat niet noodzakelijk alle locaties bezocht worden als tabelgrootte en stapgrootte niet relatief priem zijn.
2. Waartoe dient de efficiëntie-analyse van algorithmen? Waarop is deze gebaseerd bij interne methoden en bij externe methoden. Motiveer!
3. Stel het ADT tabel wordt geïmplementeerd met een boomstructuur, hoeveel elementen zou je dan plaatsen in een knoop indien (bespreek je keuzes)
    1. de boom volledig in intern geheugen zit.
    2. de boom volledig in extern geheugen wordt opgeslagen omdat deze te groot is.
4. Leg a.d.h.v. een figuur uit hoe je een index bestand kan gebruiken bij hashing. Geef een opsomming van de opeenvolgende stappen die nodig zijn bij het verwijderen van een element uit deze structuur.

Praktijk

1. Beschouw onderstaande rood-zwart boom.



    1. Voeg aan de boom achtereenvolgens de elementen met zoeksleutels 64, 25, 84, 52, 47 toe en geef telkens resulterende rood-zwart boom weer.
    2. Verwijder van de boom bekomen door vorige operaties het element met zoeksleutel 30, en geeft de rood-zwart boom weer.
2. Geef een concreet voorbeeld van een 2-3 boom met hoogte 4 waarbij door het verwijderen van één enkel element de hoogte 3 wordt, zorg er bovendien ook voor dat de boom waar je mee vertrekt zoveel mogelijk elementen bevat en dat het element dat je verwijdert niet in een blad zit.
3. Veronderstel ADT Stack. Schrijf in pseudo-code een operatie voor dit ADT waarbij elk voorkomen van een gespecifieerd item uit de stack verwijdert wordt, terwijl de volgorde van

de andere items gerespecteerd blijft. Maak hiervoor enkel gebruik van de ADT operaties.

Labotoets

1. Vertrek van je reeds geschreven ADT binaire zoekboom. De unieke zoeksleutels zijn van het type INTEGER. Voeg er de volgende recursieve operaties aan toe:
    1. Geef het item met de grootste zoeksleutel weer.
    2. Bepaal het hoogste niveau van de boom dat vol is (dat met andere woorden het maximale aantal knopen bevat).
2. Voer volgende reeks operaties uit:
    1. Creëer een lege binaire zoekboom.
    2. Voeg achtereenvolgens 6,8,15,3,9,7,14,1,10,4 en 13 toe.
    3. Verwijder hieruit achtereenvolgens 14,8 en 6.
    4. Voeg hieraan 11 en 12 toe.
    5. Bepaal de hoogte van de binaire zoekboom.
    6. Geef het element met de grootste zoeksleutel weer.
    7. Geef weer tot welk niveau de boom vol is.
3. Implementeer het ADT tabel, d.m.v. hashing. De unieke zoeksleutels zijn van het type INTEGER. Gebruik een hashtabel met tableSize 7. Botsingen (collisions) worden opgevangen d.m.v. separate chaining. In plaats van enkelvoudige gelinkte lijsten, worden er binaire zoekbomen gebruikt. Voorzie ook een operatie waarmee je kan weergeven welke items er in je hashtable aanwezig zijn (met hun juiste locatie), zodat je je hashtabel kan testen.
4. Voer volgende reeks operaties uit:
    1. Creëer een lege tabel (tableSize 7)
    2. Voeg toe: 30, 51, 65, 40, 28, 7, 26, 16, 49, 70, 12, 37, 35.
    3. Toon de hashtabel.
    4. Verwijder hieruit achtereenvolgens 7, 26, 49, 30
    5. Voeg toe: 72, 17, 36, 104, 87
    6. Toon de hashtabel
5. (Enkel voor wie tijd over heeft) zorg er voor dat je hashtabel hiervoor geïmplementeerd dynamisch kan groeien. Van zodra één van de zoekbomen een hoogte groter dan " krijgt, moet de tableSize van de hashtabel aangepast worden: de nieuwe tableSize wordt het eerste priemgetal groter dan 2 keer de oude tableSize. Je mag als maximale grootte van je hashtabel 29 aannemen.

Academiejaar 2007 - 2008 - 1ste zittijd

Theorie

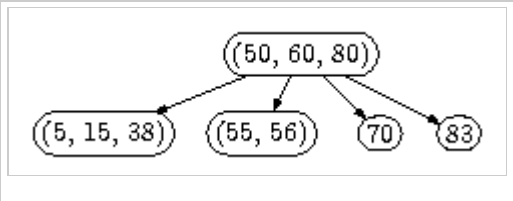
1. Geef de formele definitie van $\mathcal{O}(f(n))$. Bespreek $\mathcal{O}(1)$, $\mathcal{O}(\log n)$, $\mathcal{O}(n)$, $\mathcal{O}(n \log n)$, $\mathcal{O}(n^2)$, $\mathcal{O}(n^3)$ en $\mathcal{O}(2^n)$ op een intuïtieve manier.
2. Gegeven de verzameling van alle verplichte vakken in de bacheloropleiding, met hun noodzakelijke voorkennis als een lijst van vakken die afgelegd dienen te worden. Je kan pas voor een vak inschrijven als je alle vakken van de voorkennis afgelegd hebt. Je wenst na te gaan in welke volgorde je de verschillende vakken kan volgen. Leg uit welke toepassing van graven je hiervoor gezien hebt en hoe je te werkt gaat.
3. Wat is timedriven Simulatie & eventdriven simulatie
4. Gegeven B-boom met graad 5 en hoogte 3. Bepaal de maximum en minimum aantal records van de boom en min & max aantal knopen van de boom

Oefeningen

1. Schrijf een recursieve procedure voor het kleinste getal te zoeken van een BST (Binary Search Tree) (Pseudo code)

2. Schrijf een procedure die het aantal items van een queue telt zonder de queue te verwijderen (Pseudo code)

3. Insert 75 , 20 , 16 en 18 in de 2-3-4 boom (Fig. fig:234tree)



Zet deze 2-3-4 boom om in een Red Black tree na het inserten van deze getallen.

Academiejaar 2002 - 2003 - 1ste zittijd

Theorie

1. Geef definitie en motivatie van ADT.
2. Geef formele definitie van 2-3boom.
3. Leg het hoe en waarom uit van het indexeren van externe bestanden (mbv figuren).
4. Bespreek het optreden en behandelen van botsingen bij hashing.

Oefeningen

1. Schrijf in pseudocode een operatie voor het ADT Queue, waarbij het aantal items van een Queue geteld wordt. De Queue moet blijven bestaan, maak hierbij enkel gebruik van ADT operaties.
2. Veronderstel een binaire boom waarvan de items Integers zijn. Schrijf in pseudocode een recursieve operatie die de som van de elementen van de binaire boom berekent.
3. Oefening op gegeven B-tree (orde?, insert en delete).

Academiejaar 2000 - 2001 - 1ste zittijd

Theorie

1. Bespreek hashing en collision handling. Voor welk ADT wordt hashing gebruikt? Is hashing geschikt voor alle soorten bewerkingen op dit ADT? Waarom (niet)?
2. In welk opzicht zijn 2-3 zoekbomen beter dan binaire zoekbomen voor de implementatie van een tabel? Zijn bomen die knooppunten toelaten met meer dan 3 kinderen dan nog beter? Motiveer je antwoord.
3. Wat verstaan we onder het indexeren van een externe file en waarom wordt het ge- bruikt? Bespreek 2 manieren waarop index files kunnen georganiseerd worden.

Praktijk

1. Bij de pointer gebaseerde implementatie van het ADT stack willen we een operatie implementeren die een lege stack aanmaakt. Welke van de 4 volgende stukjes code geeft hiervoor de beste implementatie weer? Verklaar je antwoord.

1. PROCEDURE Create1 (s: stack); begin s := NIL end;
2. PROCEDURE Create2 (s: stack); begin NEW(s); s := NIL end;
3. PROCEDURE Create3 (VAR s: stack); begin s := NIL end;
4. PROCEDURE Create4 (VAR s: stack); begin NEW(s); s := NIL end;

2. Sorteer de volgende rij getallen met heapsort: 44 55 12 42 94 18 6 67. Laat bij elke stap de heap zien, zodat de werking van het algoritme duidelijk blijkt.

Academiejaar 1997 - 1998 - 1ste zittijd

Theorie

1. Bespreek de implementatie van een Heap. Waarom is deze implementatie juist zo geschikt voor Heaps? Waarvoor worden Heaps gebruikt?
2. Wat bedoelt men met collision handling? Bespreek de verschillende mogelijkheden.
3. Waarvoor is een indexrecord of indexfile nuttig? Bespreek de voordelen.

Oefeningen

1. Heapsort de volgende sequentie: 25 57 48 37 12 92 86 33.
2. Stel je hebt een priority queue geimplementeerd door binaire boom. Geef de voorstelling van de priority queue na elk van de volgende operaties (links = priority queue, midden = prioriteit, rechts = inhoud): Create(pq), Insert(pq,2,6), Insert(pq,2,7), Insert(pq,3,6), Insert(pq,1,8), Insert(pq,3,3), Remove(pq), Insert(pq,4,2), Insert(pq,6,2),Remove(pq), Remove(pq).

Academiejaar 1995 - 1996 - 1ste zittijd

1. Geef een overzicht van de verschillende datastructuren en bespreek telkens de mogelijke implementaties.

Academiejaar 1994 - 1995 - 1ste zittijd

1. Bespreek het algoritme dat de infix expressie evalueert. Van welke ADT(s) maak je gebruik?
2. Geef de definities van tabellen en bomen. Verband?
3. Schrijf een programma voor “topological sorting”. Gebruik recursie.
4. Toon aan dat de delete operatie van 2-3bomen correct werkt.

Academiejaar 1994 - 1995 - 2de zittijd

1. Definitie: ADT + leg uit: position oriented en value oriented + voorbeelden.
2. Definitie heap.
3. Leg uit: Hashing.

Oud examenvoorbeeld

1. Definieer het ADT Stack. Is een procedure StackTop(S,e): kopieer topelement van S met
   - preconditie: S bestaat
   - postconditie: S onveranderd, e = en

vereist? Hoe is de vergelijkbare implementatie bij Queue's?

2. Definieer binaire bomen. Bespreek de verschillende implementatiemogelijkheden. (Geef telkens type declaraties geassocieerd met de voornaamste datastructuur)

Ontvangen van "http://www.winak.be/tuyaux/index.php?title=Gegevensstructuren"
Categorieën: Informatica | 1BINF

Inleiding Programmeren

Uit Encyclopedia Academia

Bespreking

Inleiding Programmeren	
Professor	Frans Arickx
Richting	Informatica
Jaar	1BINF

Vermits we vanaf 2011-2012 niet meer werken met Oberon maar met Python en de prof veranderd is, is de onderstaande informatie niet echt van toepassing.

Er zijn bij dit vak twee soorten studenten: diegenen die niet kunnen programmeren en vinden dat ze in de lessen te snel gaan, en diegenen die al wel kunnen programmeren (niet meteen in Oberon) waarvoor de lessen wat langzaam gaan.

Toch ligt het slaagpercentage bij dit vak behoorlijk wat lager dan dat men zou verwachten. De cursus beoogt 2 belangrijke aspecten: het aanleren van programmeren en het geven van een inleiding in programmeertechnieken die je in latere vakken in meer detail zult behandelen. Dit alles gebeurt op een formele manier met extra aandacht voor de formele syntactische beschrijving van de taal (EBNF). Het is dit laatste dat voornamelijk van belang is bij het theoretische gedeelte van het vak terwijl het eerste deel van belang is bij de praktijk.

Theorie

Bij het theoretisch gedeelte van dit vak is het een kwestie van de verschillende programmeertechnieken goed te kennen op een formele manier en de EBNF goed te begrijpen. De professor heeft graag dat je veel uitleg schrijft (maar geen overbodige uitleg), dus wees niet te beknopt in je antwoorden. Als je moeilijkheden hebt met het begrijpen van de theorie staan er in de bibliotheek enkele boeken die de lastigere delen van de cursus beter documenteren.

Ook moet je een mondeling examen afleggen, maar indien je de stof goed beheerst is dit een rustig babbeltje met de prof.

Elk jaar komt er trouwens een vraag op het schriftelijk examen in verband met de runtime stack. De werking hiervan werd uitgelegd tijdens de les, maar staat nergens in de cursus. Wikipedia (http://en.wikipedia.org/wiki/Call_stack) heeft een uitgebreid artikel in verband met dit onderdeel.

Praktijk

Aan de praktijk mispakken veel studenten zich: je krijgt namelijk 3 uur om een programmeeropdracht uit te voeren en dit is relatief weinig. Bijna altijd is de programmeeropdracht een opgave waarin een gelinkte lijst moet gemaakt worden van een andere gelinkte lijst, die op zijn beurt een gelinkte lijst bevat, enzovoorts. Om voor elke lijst opnieuw met pointers te werken en hiermee te knoeien is het veel eenvoudiger op voorhand een goed uitgewerkte gelinkte lijst te maken op lege nodes, het is namelijk toegelaten om eigen programmatuur mee te nemen. Kijk naar het voorbeeldje van professor Arickx zijn OOLists om dit te doen en zorg ervoor dat je lijst Nodes kan toevoegen, verwijderen, vervangen, …. Als je dan tijdens het examen bijvoorbeeld een lijst van CD's moet maken kun je gewoon een klasse CD overerven van de klasse node die je even gemakkelijk als een gewone node kunt toevoegen aan de lijst, zonder deze telkens opnieuw te schrijven. Het kan natuurlijk wel zijn dat je ook speciale bewerkingen moet kunnen toepassen op de lijst zoals het zoeken van een CD van een bepaalde auteur. Dit doe je dan door je lijst over te erven van de standaardlijst en hierop deze bewerkingen uit te voeren. Door dit te doen heb je al

zeker een voorsprong opgebouwd in snelheid en in mooie programmeerstijl wat ook zeker geapprecieerd wordt.

Ga ook zeker de eerste praktijkles bijwonen aangezien dan de programmeeromgeving wordt uitgelegd en dit niet echt vanzelfsprekend is tegenover 'normale' programmeeromgevingen.

Puntenverdeling

Theorie: 10/20. Praktijk: 10/20.

Examenvragen

Academiejaar 2010 - 2011 - 1ste zittijd

1. Bespreek het concept "array" in al zijn facetten
2. Bespreek het concept "pointer" in al zijn facetten
    1. Geef de ouput van onderstaand programma
    2. Teken de run-time stack zo duidelijk mogelijk (annoteer met commentaar waar nodig) net voor de finale terugkeer uit Magic bij de toekenning van z[0] in LoopArray

```
Module Examen;
Import OutExt;

PROCEDURE Magic(a: INTEGER; b: LONGINT): LONGINT;
BEGIN
   IF((a - b) < 1) THEN
      RETURN b;
   ELSE
   RETURN b + Magic(a, b + 2);
   END;
END Magic;

PROCEDURE LoopArray(k: INTEGER; VAR z: ARRAY OF CHAR);
   VAR
      i: LONGINT;
BEGIN
   FOR i := 0 TO LEN(z) - 1 DO
      z[i] := CHR(Magic(k, i) + ORD('a'));
   END;
END LoopArray;

PROCEDURE Doe*;
   VAR
      i: INTEGER;
      a: ARRAY 2 OF CHAR;
BEGIN
   LoopArray(1, a);
   FOR i := 0 TO 1 DO
      OutExt.Char(a[i]);
   END;
   OutExt.Ln;
END Doe;

BEGIN
END Examen.
```

Op het mondelinge examen kreeg een student de volgende vraag:

1. Wat is het nut en het verschil van en tussen het statisch type en het dynamisch type van een pointervariabele.

Academiejaar 2009 - 2010 - 1ste zittijd

Theorie

1. 1. Geef een overzicht van de samengestelde types, en de mogelijkheden die ze bieden. Behandel ook overeenkomsten en verschillen tussen deze types.
   2. Geef het concept, gebruik en nut van pointers bij deze samengestelde types.
2. Verklaar het begrip aparte compilatie, alsook Oberon's gescheiden compilatie. Geef specifiek de inhoud van de symbol file, en waarom, wanneer en welke instantie een deel van de inhoud gebruikt. Geef ook of dit tijdens compile of run-time was.
3. 1. Geef het verschil tussen een object en een ADT.
   2. Verklaar de begrippen statisch en dynamisch object, en het verschil hiertussen.
   3. Wat zijn de redenen en het gebruik van type test en type guard.
   4. verklaar polymorfisme, en het verband met vraag 3.3
4. Bij de Oberon run-time stack speelt het BP register een belangrijke rol. Bespreek op zo duidelijk en volledig mogelijke manier waarom deze BP nuttig en noodzakelijk is bij Oberon code generatie, en hoe hij daartoe door de compiler wordt aangewend. Verduidelijkingen door expliciete voorstellingen van de run-time stack kunnen bij de uitleg nuttig zijn.

Academiejaar 2009 - 2010 - 2de zittijd

Theorie

1. Bespreek alle aspecten van open array.
2. Bespreek de informatie uitwisseling bij procedure-oproepen
3. 1. Wat is het verschil tussen een object en een ADT
   2. Wat is het verschil tussen een statisch en een dynamisch type van een record?
   3. Wat is het nut en gebruik van type-test en type-guard?
   4. Wat is polymorfisme en geef het verband met 3.3
4. Vraag over de run-time stack, in verband met allocatie bij variabelen in geneste procedures.

Academiejaar 2008 - 2009 - 1ste zittijd

Theorie

1. Programmeertalen hebben hun eigen syntax (grammatica) en semantiek (betekenis woordenschat); zo ook Oberon.
   1. Bespreek hoe de syntax gespecificeerd wordt.
   2. Bespreek hoe (en waar) de semantiek gespecificeed wordt, en geef duidelijk aan wat de eventuele relatie met de syntaxbeschrijving is; geef hiervan eventueel enkele voorbeelden.
2. OBeron kent het concept "type extensie".
   1. Leg dit gedetailleerd uit, syntactisch en semantisch, en geef aan wat het voordel en nut ervan is.
   2. Indien er extra compatibiliteitseigenschappen voorzien zijn, bespreek deze dan.
3. Bespreek recursie op gedetailleerde wijze:
   1. Leg de basisprincipes uit, en stel de verschillende recursievoorwaarden op.
   2. Geef duidelijk weer wat de programmatorische vereisten zijn om recursie te kunnen realiseren, en geef daarna de daaraan voldoende syntactische constructie die recursie in Oberon mogelijk maakt.
   3. Geef duidelijk aan wat de verschillen tussen recursie en (klassieke lus-) iteratie zijn.
   4. Leg uit welk soort parameter (waarde of variabele) het meest geschikt is om de relevante informatie van recursiestap naar recursiestap door te geven (denk bv. aan de recursieve implementatie van de Faculteit (n!), met relevante waarde n), en waarom:

wees dus heel duidelijk als je de ene vorm boven de andere verkiest!
5. Bepsreek kort QuickSort, en bespreek omstandig de recursieve eigenschappen (implementatie + wijze van voldoen aan de recursievoorwaarden).

4. Bespreek duidelijk de opbouw van de run-time stack bij oproep van een procedure:
    1. Geef aan welke informatie, in welke volgorde, door welke verantwoordelijke, en waarom, op de stack wordt geplaatst bij oproep.
    2. Leg uit wat de noodzaak en het nut is van de Base Pointer (BP), en welke “Oberon zichtbaarheidseigenschappen” hiermee bovendien gerealiseerd worden, en waarom.

Academiejaar 2007 - 2008 - 1ste zittijd

Theorie

1. Bespreek in volledig detail:
    1. De formele hoofding van een proceduredeclaratie.
    2. De actuele oproep van een procedure.
2. Leg in volledig detail uit:
    1. De noodzaak aan pointers, en hoe ze in Oberon aangewend worden voor dynamisch geheugengebruik.
    2. De voordelen van, en de gevaren bij, het gebruik van pointers.
3. Leg in volledig detail uit:
    1. Het verschil tussen een object en een klassiek ADT.
    2. Het verschil tussen het statisch en dynamisch type van objecten.
    3. De redenen voor, en het gebruik van, type-test en type-guard.
    4. Het concept polymorfisme, en het verband met onderdeel (c).
4. Bespreek duidelijk de opbouw van de run-time stack bij oproep van een procedure met zowel waarde en variabele parameters, als lokale variabelen. Geef in detail aan welke de verantwoordelijkheden zijn van de oproepende en opgeroepen procedure, en wat de rol is van de SP en de BP hierbij.
5. (Mondeling) Leg de globaliteit en zichtbaarheid uit binnen een module. (Hint: Absoluut globaal [In module: Const, Type, Var, Buitenste Procedure], globaal, lokaal [Enkel in procedure zelf])
6. (Mondeling) Zijn de dingen die absoluut globaal zijn buiten de module beschikbaar? (Hint: Exporteren).
7. (Mondeling) Leg het geheugenbeheer (Hint: Run-time stack!!!) uit van globale variabelen (Hint: BS, Stack Peeking)
8. (Mondeling) Waar bevinden de geexporteerde absoluut globale variabelen zich in het geheugen en waarom? (Hint: Heap, omdat je vanuit andere modules niet kan Stack Peeken)
9. (Mondeling) Leg de compilatievolgorde uit.
10. (Mondeling) Vertel wat over samengestelde types.

Academiejaar 2007 - 2008 - 2de zittijd

Theorie

1. Recursie: leg uit, geef de voorwaarden en bespreek Quicksort aan de hand van recursie.
2. Welke samengestelde types zijn er, leg zo compleet mogelijk uit.
3. In Oberon spreekt men van ``gescheiden compilatie". Leg uit.
4. Wat is de SP en BP en geef wat uitleg over het gebruik ervan.
5. (Mondeling) Wat weet je over OO?

6. (Mondeling) Leg polymorfisme uit.

Academiejaar 2006 - 2007 - 2de zittijd

Theorie

1. Wat is het nut van OO programmeren en wat voorziet Oberon om OO te ondersteunen? Wat is het verschil met procedureel programmeren?
2. Leg alle lussen uit die mogelijk zijn in Oberon. Geef een voorbeeld wanneer elke lus het best kan gebruikt worde + leg uit waarom de andere lussen hier minder voor geschikt zijn.
3. Run-Time Stack: hoe werkt het? Leg gedetailleerder uit welke rol de BP en SP hierbij hebben.
4. Waar liggen de syntactische en semantische regels opgeschreven? Wat is het verschil? Leg zo volledig mogelijk uit. (deze vraag was helemaal niet zo geformuleerd, maar tkwam erop neer ...)

Academiejaar 2004 - 2005 - 1ste zittijd

Theorie

1. Bespreek de noodzaak en nu van een éénduidige syntaxbeschrijving voor Oberon-2, en hoe deze gespecifieerd kan worden. Welke plaats nemen de $uitvoeringsbesluiten$ (de semantische aanvullingen) hierbij in, en waar vindt men deze terug? Geef een paar voorbeelden.
2. Bespreek de eigenschappen van het $boolean$ type en de bijhorende logische operatoren. Wat is hun toepassing bij de controle-instructies?
3. Bespreek heterogeen samengestelde types, en hun specifieke gebruikseigenschappen als parameter. Wees zo volledig mogelijk!
4. Er werden in Oberon-2 een aantal syntactische communicatiemogelijkheden naar en van de procedures vastgelegd via de formele hoofding. Bespreek ale mogelijkheden, en geef zo volledig mogelijk de voorkomende compatibiliteitsregels weer met uitleg.
5. Welke zijn de syntactische mogelijkheden die Oberon-2 voorziet voor *object-georiënteerd programmeren*? Geef zo volledig mogelijk weer welke bijkomenende voordelen deze uitbreidingen meebrengen. Geef eventueel een paar voorbeelden.
6. $(enkel\ voor\ de\ informatica)$ Bespreek op gestructureerde wijze wat er op de run-time stack wordt geplaatst bij de oproep en uitvoering van een procedure. Geef voldoende uitleg bij elk van de verschillende onderdelen.

Academiejaar 2004 - 2005 - 2de zittijd

Theorie

1. Bespreek zo volledig mogelijk de voorkomende vormen van formele procedure hoofdingen, en de bijhorende toepassingsmogelijkheden in Oberon. welke plaats nemen $compatibiliteitseigenschappen$ hierbij in?
2. Geef een gedetailleerd en volledig overzicht van de toepassingsmogelijkheden van homogeen samengestelde types in Oberon
3. Geef duidelijk weer wat polymorfisme betekent, en wat de voordelen ervan zijn bij het programmeren. Geef ook duidelijk aan op welke wijze het in Oberon toegepast kan worden (syntactische cancepten, mogelijke voorwaarden, …)
4. $(enkel\ voor\ de\ informatica)$ De code maakt op de run-time stack gebruik van de $base-pointer\ (BP)$. Leg omstandig en met grafische voorstelling van de run-time stack, uit

waartoe de base pointer in deze context dient.

Academiejaar 2003 - 2004 - 1ste zittijd

Theorie

1. 1. In de EBNF-beschrijving ontbreken een aantal specificaties om foutloze compilatie mogelijk te maken. Over welke specificaties gaat het, en waar vindt men deze terug?
   2. Bespreek de ontbrek hd specificaties expliciet bij (een EBNF-syntax kaart is achteraan toegevoegd):
      1. de PROCEDURE declaratie
      2. de CASE instructie
      3. de assignment (":=")

1. Bespreek de lus instructies, en hun eigenschappen. Geef in je eigen woorden weer in welke omstandigheden je een bepqalde lusstructuur prefereert, en waarom (je kan voorbeeldsituaties gebru1ken)
2. Bespreek de homogeen samengestelde types, en hun gebruikseigenschappen als parameter.
3. ## Bespreek alle compatibiliteitsregels die verband houden met de parameters van een procedure, en waarom ze van belang zijn.
   1. Bespreek op ondubbelzinnige wijze wanneer je een parameter van een pointer type als waarde of variabele parameter declareert.
4. De object-georiënteerde extensies in Oberon-2 geven het begrip ADT bij het programmeren extra mogelijkheden. Probeer zo volledig en duidelijk mogelijk aan te geven wat de voordelen zijn, voor zowel de ontwikkelaar van het ADT als de gebruiker ervan. Indien je echter de klassieke ADT variant verkiest, leg dan duidelijk uit waarom.
5. Van welke soorten "typering" mag een receiver parameter zijn? Waarom zijn volgens u enkel de door u opgesomde soorten toegelaten, en is er een verband tussen oe verschillende mogelijkheden?

Academiejaar 2003 - 2004 - 2de zittijd

Theorie

1. Oberon heeft een gevarieerde woordenschat. Geef de syntax voor een "woord" op, en bespreek alle mogelijke semantisch verschillende betekenissen ervan. Geef duidelijk weer waar de semantiek van elk soort wordt vastgelegd, en welke instantie "verantwoordelijk" is voor de betekenis; hoe lang geldt voor elke soort de levensduur?
2. Bespreek de samengestelde typeconstructies in Oberon. Bespreek hun toepassIng binnen het programmeren in Oberon. Vergelijk ze met elkaar, en geef de respectievelijke voor- en nadelen weer.
3. Bespreek recursie. Als je informatie wenst door te geven naar de volgende recursiestap, welke soort informatiedoorgave is dan het meest geschikt en waarom?
4. Bespreek de syntactische elementen die noodzakelijk (en voorhanden!) zijn voor object-georiïnteerd programmeren in Oberon. Bespreek in het bijzonder de eigenschappen van de receiver parameter (o.a. de typering ervan).
5. *(enkel voor de informatica)* Bespreek gedetailleerd de inhoud van de run-time stack.

Academiejaar 2002 - 2003 - 1ste zittijd

Theorie

1. Bespreek de declaratie van een procedure
    1. Geef voor elk syntactisch onderdeel van de declaratie aan wat de betekenis en bedoeling ervan is. Wees volledig!
    2. Horen er, naast de EBNF-syntax rege1s, ook nog semantische regels (dus niet vervat in de EBNF-syntax) waarop de compiler fouten kan genereren? Zo ja, geef een zo volledig mogelijk overzicht van deze elementen, enkel voor de declaratie van een procedure.
2. Bespreek de communicatiemogelijkheden van en naar procedures, eh geef de eigenschappen van elke mogelijkheid weer. Bespreek ook de rol die de compatibiliteitseigenschappen spelen.
3. Bespreek zo volledig mogelijk de keuze-instructies.
4. Geef bij volgende declaraties:

```
TYPE
    Ptr = POINTER TO Rec;
    Rec =
     RECORD
        a : REAL;
        b : Ptr;
     END;
VAR
    p1, p2: Ptr;
```

   zo duidelijk mogelijk aan wat het verschil is tussen “p1 := p2” en “p1↑ := p2↑”.

5. In de opbouw van ADT’s vormen de “Klassen” (typering voor object-veranderlijken) een ultieme implementatiemogelijkheid. Geef aan waarom hierbij het object-concept “beter” is, m.a.w. bespreek gebruiksgemak, beveiliging, uitbreidbaarheid, leesbaarheid, …

Academiejaar 2002 - 2003 - 2de zittijd

Theorie

1. In Oberon programma’s komen declargtiedelen voor:
    1. Geef aan “waar” deze overal kunnen voorkomen
    2. Som alle mogelijke verschillende soorten declaraties’s op
    3. Geef voor elke soort de noodzaak weer; m.a.w. geef aan waarom deze declaratie voor de compiler noodzakelijk is bij het compileren van de code
    4. Zijn er declaraties die ook buiten het gecompileerde programma beschikbaar zijn? Zo ja, hoe worden ze beschikbaar gesteld, en gebruikt
2. Bespreek nut en de noodzaak van procedures. Wees volledig over de toepassingsmogelijkheden ervan, en bespreek daarbij de mogelijke syntactische vormen.
3. Bespreek zo volledig mogelijk de lus-instructies.
4. Klassen en objecten geven, naast een andere vorm dan het procedurele programmeren, bijkomende mogelijkheden aan een programmeur om sne11er (?), Beter (?), onderhoudsvriendelijker (?), … te programmeren. Bespreek dit zo volledig mogelijk (dus met inbegrip van de bijkomende syntactische constructies), en je eigen mening telt mee!

Academiejaar 2001 - 2002 - 1ste zittijd

Theorie

Schriftelijke proef

1. 1. De volledige syntaxdefinitie van Oberon(-2) is samengevat in EBNF-vorm. Nochthans is dit onvoldoende voor de compiler om een programma op syntactische correctheid te testen. Waarom? Wat is dan de betekenis van de extra nodige informatie, en waar staat deze beschreven?
   2. Leg duidelijk uit wat in de syntax van Oberon bedoeld wordt met “Gereserveerde woorden (Reserved)”, “Gepredefinieerde woorden (Predefined)” en “Gebruikersgedefinieerde woorden (User-defined)”; gebruik voldoende voorbeelden ter illustratie van de overeenkomsten/verschillen.
2. Bespreek alle voorkomende vormen van “toekenningscompatibiliteit” en leg telkens uit waarom de regel werd ingevoerd. Wees volledig!
3. *(enkel voor de informatica)* Bespreek de recursievoorwaarden, en illustreer aan de hand van QuickSort.
4. Leg in eigen woorden uit waarom “OO-abstractie” “beter” is dan “Klassieke Data-abstractie”, Zijn er ook nadelige aspecten aan?
5. Een MODULE lijkt in een aantal opzichten op een object/klasse. Welke gelijkenissen (expliciet zowel als impliciet) herken je, en waar zijn de verschillen?
6. *(enkel voor de informatica)* De “receiver” parameter moet ofwel een waarde parameter “pointer to record” ofwel een variabele parameter “record” zijn. Leg uit of bedenk een zinnige reden hiervoor (eventuele hint: denk aan de wijze waarop parameters in de run-time stackopbouw doorgegeven worden).

Mondelinge proef

We kregen een van de volgende vragen

- Bespreek de communicatie tussen procedures
- Bespreek gescheiden compilatie
- Bespreek de stack
- Bespreek de interne voorstelling van een floating pointgetal IEEE.

```
Wat is polymorfisme?
```

Academiejaar 2001 - 2002 - 2de zittijd

Theorie

1. Bespreek de informatiedoorgave van en naar Procedures via parameters. Geef aan waar de verschillende soorten parameters voor staan in functie van een veilige programmeertoepassing. Geef ook aan welke vorm het type van de formele parameter kan aannemen (dus: welke types kunnen allemaal voorkomen).
2. 1. Bespreek alle “compatibiliteits-eigenschappen” die in Oberon voorkomen, en in welke omstandigheden ze voorkomen (syntactische constructies). Geef telkens aan waarom het zin heeft dat deze compatibiliteitseigenschappen werden ingevoerd.
   2. Zijn er compatibiliteitseigenschappen die potentieel “gevaarlijk” zijn, dus onverwachte resultaten kunnen opleveren? Zo ja, geef van elk mogelijk gevaar een voorbeeld.

3. *(enkel voor de Informatica)* Bespreek het concept Polymorfisme, en hoe het in Oberon voorkomt en wordt toegepast.

```
{}

(enkel voor de Wiskunde) Bespreek het concept Type-extensie en zijn toepassingsmogelijkheden.
```

Academiejaar 2000 - 2001 - 1ste zittijd

Theorie

1. De volledige syntaxdefinitie van Oberon(-2) is samengevat in EBNF-vorm. Is dit voldoende voor de compiler om een compilatie-eenheid op correctheid te testen en machinecode te genereren? Zo niet, wat is er meer nodig en waarom?
2. Oberon(-2) kent 3 soorten identifiers. Som ze op en geef aan waarvoor ze dienen.
3. Compatibiliteitsregels geven soms aanleiding tot “versoepeling” van de type-checking. Som zo volledig mogelijk de in Oberon(-2) aanwezige compatibiliteitsregels op, en geef telkens aan waarom de “versoepeling” nuttig of belangrijk is.
4. Veronderstel een globale variabele met identifier “naam”, en een lokale variabele van een procedure, eveneens met identifier “naam”.
    - Mag dit?
    - Als het mag, moeten ze dan van hetzelfde type zijn?
    - Als het mag, in welke situatie wordt elke variabele gebruikt, en wat gebeurt er op dat oqenblik met de andere?
5.
    - Geef in een tabelvorm alle ITEM-“verantwoordelijkheden” weer voor elke lusvorm.
    - Zou ITEM ook bij recursie verantwoordelijkheden kunnen vastleggen, en zo ja, hoe liggen ze dan?
6. Wat is het onderscheid tussen “procedurele”-, “data”- en “object-georiënteerde”- abstractie?
7. Wanneer gebruikt men voor pointers een `‘waarde’ en wanneer een “variabele” parameter?
8. Geef een duidelijke motivatie voor het correct of niet-correct zijn van volgende uitspraak: “het is aangewezen de creatie van nieuwe objecten via NEW uit te voeren binnen de module/procedure waar de declaratie van het object voorkomt”.
9. Leg op duidelijke wijze het onderscheid uit tussen “polymorfisme” en het gebruik van de “type-test + type-guard” methodiek. Welke is best in welke omstandigheid?

Academiejaar 2000 - 2001 - 2de zittijd

Theorie

1. In Oberon modules worden declaratie- en instructiegedeeltengebruikt. Bespreek waar deze delen voorkomen samen met de eigenschappen die ze vertonen (nut, zichtbaarheid, …). Maak een volledig en sluitend verhaal.
2. Leg uit hoe object-oriëntatie de abstractiemogelijkheden in Oberon optimaliseert, en dit bv. in contrast met de klassieke procedurele of data abstractie.
3. Bespreek de recursievoorwaarqen, en leg aan de hand van QuickSort duidelijk uit hoe en waar ze gerealiseerd worden.

Academiejaar 1999 - 2000

Theorie

1. De volledige syntaxspecificatie van Oberon(-2) is samengevat in EBNF-vorm. Is dit

voldoende voor de compiler om een compilatie-eenheid op correctheid te testen en machinecode te genereren? Zo niet, wat is er meer nodig en waarom?
2. Compatibiliteitsregels geven soms aanleiding tot “versoepeling” van de type-checking. Som zo volledig mogelijk de in Oberon(-2) aanwezige compatibiliteitsregels op, en geef telkens aan waarom de “versoepeling” nuttig of belangrijk is.
3. Veronderstel een globale variabele met identifier “naam”, en een lokale variabele van een procedure, eveneens met identifier “naam”.
    1. Mag dit?
    2. Als het mag, moeten ze dan van hetzelfde type zijn?
    3. Als het mag, in welke situatie wordt elke variabele gebruikt, en wat gebeurt er op dat ogenblik met de andere?
4. Wat is het onderscheid tussen “procedurele”-, “data”- en object-georiënteerde- abstractie?
5. In de cursus programmeren staat het concept Abstract Data Type (ADT) eigenlijk centraal, en dienen de meeste behandelde onderdelen om dit concept qua opbouw, maar ook qua beveiliging, ultiem te ondersteunen. Geef voldoende omstandig definitie, nut, toepassingsgebied en voordelen (eventuele nadelen) weer van een ADT. Leg voldoende omstandig uit op welke wijze elk van de volgende trefwoorden (eventueel) met ADT’s te maken hebben, en/of hoe ze het concept helpen realiseren; bedenk de betekenis van “… qua opbouw, maar ook qua beveiliging …” uit de inleidende zin hierbij:
    1. Type (algemeen)
    2. Samengestelde types
    3. Procedures
    4. Parameters
    5. Type-compatibiliteitseigenschappen
    6. Modules
    7. Pointers
    8. Controlestructuren
    9. Objecten
    10. Polymorfisme
6. Recursie is een belangrijke algoritmische oplossingstechniek. Bespreek de “recursievoor-waarden” opdat een probleem zo kan worden opgelost, geef gedetailleerd weerd hoe in dat opzicht QuickSort een sorteeroplossing biedt - geef in het bijzonder duidelijk aan waar de recursievoorwaarden aan bod komen - en vergelijk en evalueer recursie t.o.v. iteratie.
7. “Objecten” nemen een belangrijke plaats in in het moderne programmeren. Zo heeft ook Oberon(-2) dit concept voorzien in de syntax. Bespreek de syntactische specificaties voor “compile-time bound procedures”, van welk type de betrokken ob ject-parameters mogelijk zijn en waarom, en hoe dit aanleiding geeft tot “klasse-eigenschappen” zoals bv. Polymorfisme.

Academiejaar 1998 - 1999 - 1ste zittijd

Theorie

- In Oberon, net als in veel andere programmeertalen, zijn de begrippen “declaratie” en “declaratiegedeelte” zeer belangrijk. Bespreek ze op een volledige doch gestructureerde wijze.

Academiejaar 1995 - 1996 - 1ste zittijd

Theorie

1. Leg uit hoe u een programmeeropgave tot een oplossing brengt, en geef de redenen aan

waarom u deze methodiek volgt. Zou deze manier gebruikt kunnen worden voor zowel procedureel als ob jectgeorinteerde programmeren? Motiveer uw antwoord voldoende.
2. Bespreek zo volledig mogelijk het nut van procedures. Leg het verband tussen de actuele oproep en formele declaratie; leg hierbij uit hoe de compiler via de stackstructuur het verband legt tussen de actuele en formele parameters.
3. Wat is het verband tussen data-abstractie en objectoriëntatie? Zijn er voordelen om data-abstractie via modulaire technieken te implementeren, of zijn modulariteit en type-extensie complementair? Geef uw antwoord voldoende duidelijk weer.
4. Geef alle sorteeralgoritmes weer die eenzelfde tijdscomplexiteit hebben, doch orden ze telkens naar specifieke tijdscomplexiteit (met verantwoording) bij de input van:
    - een reeds gesorteerde array
    - een omgekeerd gesorteerde array
    - een array met volgende initile ordening: grootste, kleinste, 2e grootste, 2e klein- ste,…

Academiejaar 1995 - 1996 - 2de zittijd

Theorie

1. Bespreek de betekenis en de mogelijke inhoud van het declaratiegedeelte van mod-ules/procedures zo volledig mogelijk.
2. Wat is het nut en de bruikbaarheid van globale veranderlijken in oberon modules? Som de mogelijke voordelen en nadelen op. Denk ruim!
3. In welke programmeeromstandigheden gebruikt men het Pointertype? Leg voldoende omstandig uit. Zijn er gevaren aan verbonden, zo ja dewelke, en waarom opteert men dan toch voor de Pointerkeuze?
4. Geef het verband tussen ADT's en objecten.

Academiejaar 1994 - 1995 - 1ste zittijd

Theorie

1. Wat vormen de karakteristieken van een TYPE? Geef hierbij via een voorbeeld bij een gepredefinieerd, alsook bij een zelf-gedefinieerd type, al deze karakteristieken voldoende gedetailleerd weer.
2. Bespreek de concepten "zichtbaarheid" en "bereikbaarheid" van gedeclareerde instanties. Hoe staan deze in verband met de begrippen "lokaliteit" en "globaliteit" ? Bestaan deze buiten de begrenzingen van individuele modules?
3. Waarom is oberon "bedrijfszeker" bij het gebruik van elementen uit andere modules via IMPORT-lijsten? Welke controles worden hiervoor uitgevoerd, hoe, wanneer en door welke instantie?
4. Bespreek data-abstractie, en de implementatiemogelijkheden ervan in oberon. Geef de verschillende respectievelijke voor- en nadelen van de verschillende implementatietech-nieken weer.
5. Geef overeenkomsten en verschillen tussen ob jecten met "methodes" en met een "bood-schappenverwerker". Welke draagt uw voorkeur weg en waarom?

Praktijk

Academiejaar 2010 - 2011 - 1ste zittijd

Opdracht

Maak een implementatie van het spel MasterMind. Het spel gaat als volgt: er wordt een combinatie van k kleuren gekozen door het programma (uit n verschillende kleuren). De gebruiker van het programma moet zo snel mogelijk de code proberen te kraken, met een maximum van p pogingen.

Basisversie

In de basisversie volstaat volgende functionaliteit:

- Er kan gekozen worden uit 8 kleuren: wit, zwart, rood, blauw, oranje, geel, groen en bruin.
- Een kleurencombinatie bestaat steeds uit 4 kleuren, het maximum aantal pogingen is 10.
- Zorg ervoor dat de kleurencombinatie willekeurig door de computer gekozen wordt (eenzelfde kleur mag meermaals in de code voorkomen!). Gebruik de module RandomNumbers.
- Voorzie een commando Cheat, die de winnende kleurencombinatie afdrukt.
- Voorzie een commando Guess, waarmee de speler een poging kan ingeven. Na elke poging van de gebruiker wordt er feedback gegeven door het programma: ofwel staat de kleur op de juiste plaats (aanduiden met +); ofwel komt de kleur wel in de code voor, maar staat ze niet op de juiste plaats (aanduiden met -); ofwel komt de kleur helemaal niet in de code voor (geen aanduiding).
- Zorg ervoor dat alle vorige pogingen zichtbaar blijven. Hou dus een geschiedenis van de vorige pogingen bij.
- Druk na beëindiging van het spel (hetzij doordat het maximum aantal pogingen bereikt is, hetzij doordat de code doorbroken is) een passende booschap af.
- Zorg ervoor dat op elk moment een nieuw spel kan begonnen worden.

Uitbreidingen

Om extra punten te verdienen, kunnen volgende uitbreidingen toegevoegd worden aan je programma. Meer sterren betekent meer punten, maar ook meer benodigde tijd!

- (*) Het aantal pogingen p is variabel, en wordt bij aanvang van het spel ingelezen (als parameter van het commando Start). Als $p \leq 0$ is het aantal pogingen onbeperkt.
- (*) De lengte k van de kleurencombinatie is variabel, en wordt bij aanvang van het spel ingelezen (als parameter van het commando Start).
- (*) De gebruikte kleuren in het spel zijn variabel, en worden voor aanvang van het spel ingelezen met het commando:

```
Mastermind.SetColors "WI" "ZW" "RO" "BL" "OR" "GE" "GR" "BR" ~
```

- (*) Maak het spel moeilijker door in de feedback na elke poging enkel weer te geven hoeveel kleuren er op de juiste plaats en hoeveel kleuren niet op de juiste plaats voorkomen, zonder de positie van de juiste kleuren daarbij te verklappen.
- (**) Voorzie een commando Undo dat je kan aanroepen gedurende het spel en die de laatste poging ongedaan maakt.
- (**) Schrijf het geheel door gebruik te maken van OO.
- (***) Voorzie een commando om de huidige spelsituatie op te slaan in een bestand (zodat je op een later tijdstip het spel kan verder spelen) alsook een commando om een opgeslagen spelsituatie in te lezen van een bestand. Commando's

```
Mastermind.Savefile "filename.txt" ~
```

```
Mastermind.LoadFile "filename.txt" ~
```

Academiejaar 2009 - 2010 - 1ste zittijd

Opdracht

Je maakt een *parser*, een programma dat ingevoerde tekst omzet in een datastructuur, die vanuit een bestand een doorlopende tekst kan inlezen en die de gebruiker van het programma toelaat om allerlei statistieken omtrent the tekst op the vragen en bewerkingen op de tekst uit te voeren. De tekst wordt in het geheugen opgeslagen in de vorm van *zinnen*, die op hun beurt dan weer gevormd worden door een opeenvolging van *woorden*.

Basisversie

In de basisversie zijn volgende acties beschikbaar als commando's vanuit je Tool-bestand:

- Lees een tekst in vanuit een bestand met een ingelezen bestandsnaam
- Druk de eerste *n* zinnen af, waarbij *n* wordt ingelezen. Als *n* groter is dan het aantal zinnen in de tekst, wordt de volledige tekst afgedrukt.
- Lees twee woorden w_1 en w_2 in, en vervang in de tekst alle instanties van het woord w_1 door w_2.
- Druk alle zinnen af waarin een ingelezen woord minstens 2 keer voorkomt.
- Druk het *n*-de woord in de tekst af, waarbij *n* wordt ingelezen
- Druk de tekst achterstevoren af (eerst het laatste woord van de laatste zin, dan het voorlaatste woord van de laatste zin, enz.)

Volgende statistieken zijn beschikbaar als commando's vanuit je Tool-bestand:

- Geef het aantal woorden in de tekst weer.
- Geef het aantal zinnen in de tekst weer.
- Geef het aantal keer dat een ingelezen woord voorkomt in de tekst weer.
- Geef het aantal zinnen waarin een ingelezen woord minstens één keer voorkomt weer.

Een zin wordt steeds beëindigd door een punt. Hou in de basisversie van je parser geen rekening met andere leestekens. Als de zin

```
Jan, Piet en Joris riepen: ``Wij zijn dikke vriendjes''.
```

wordt ingelezen, wordt deze zin terug afgedrukt als

```
Jan Piet en Joris riepen Wij zijn dikke vriendjes.
```

Uitbreidingen

Om extra punten te verdienen, kunnen volgende uitbreidingen toegevoegd worden aan je programma.

- Schrijf het geheel door gebruik te maken van OO.

- Hou bij de berekening van de statistieken over woorden geen rekening met hoofd- of kleine letters. Het aantal keer dat het woord “Het” voorkomt is dus gelijk aan het aantal keer dat het woord “het” voorkomt.
- Stel zinnen niet alleen samen uit een reeks woorden afgesloten met een punt, maar voeg ook ondersteuning toe voor leestekens zoals komma’s, vraagtekens, uitroeptekens en aanhalingstekens.
- Voorzien één of meerdere van volgende extra acties en statistieken
  - Geef het aantal *verschillende* woorden in de tekst weer.
  - Geef het aantal woorden die met een hoofdletter beginnen weer.
  - Druk alle zinnen af waarin minstens twee maal hetzelfde woord voorkomt.
  - Druk alle woorden uit de tekst af die met een ingelezen letter beginnen.
  - Druk alle woorden uit de tekst af waarin een ingelezen substring voorkomt.
  - Druk alle woorden alfabetisch af.
  - Print de lijst van *n* meest voorkomende woorden in de tekst af, waarbij *n* wordt ingelezen.

Academiejaar 2009 - 2010 - 2de zittijd

Media:2010 InleidingProgrammeren Praktijk TweedeZit.pdf

Academiejaar 2007 - 2008 - 1ste zittijd

Afspraken

- Zet al de module bestanden die je gebruikt hebt in een archief met als naam “srolnummer.Arc”. Zoals bijvoorbeeld “s045480.Arc”.
- Gebruik enkel en alleen het Oberon lettertype voor je code (geen kleurtjes).
- Hou je strikt aan de Oberon Code Conventions.
- Je mag al de code die je ZELF hebt geschreven gebruiken.
- Je mag je cursus gebruiken.
- Elke vorm van communicatie is verboden.

Opdracht

Je maakt een elektronische versie van het gezelschap spel Zeeslag. Bij Zeeslag is het de bedoeling te ontdekken waar de vloot van je tegenspeler ligt. Dit doe je simpelweg door bommen af te vuren op de vloot van jouw tegenspeler. Om met dit spel te kunnen beginnen moet je eerst je vloot in de zee plaatsen. Dit mag horizontaal en verticaal zolang de boten elkaar maar niet direct horizontaal of verticaal raken. Concreet maak je een systeem dat 2 spelers bevat en per speler een aantal boten op zijn deel van het speelveld. Elke speler voegt om de beurt een boot toe. Het is toegestaan met ee nvast aantal boten te werken, een dynamische oplossing is natuurlijk beter en verdient meer punten, Het spel start als Speler 1 zijn eerste schot afvuurt, hierna is elke speler om de beurt aan zet.

Alle vereiste functionaliteit in volgorde van belangrijkheid. Het spreekt voor zich dat je punten afhankelijk zijn van de hoeveelheid functionaliteit die geïmplementeerd en getest is. Zorg ervoor dat er een scenario is (opeenvolging van stappen in de Tool file) die dit duidelijk toont. Vergeet ook niet om duidelijk te specificeren welke functionaliteit geïmplementeerd is.

1. In het basisprogramma mag je er vanuit gaan dat het speelveld een vaste grootte (10x10 heeft. Het systeem moet de volgende functionaliteit bevatten:

- Je moet speler 1 en speler 2 een naam kunnen geven.

- Je moet per speler een lijst van boten kunnen toevoegen met de volgende eigenschappen:
  - Naam
  - Coördinaten (controleer of ze binnen het veld liggen en niet overlappen met andere boten).
  - Elke speler moet om de beurt een schot kunnen afvuren (via de Tool file), er wordt weergegeven of een schot raak was of niet. Bij het zinken van een boot verschijnt er ee nboodschap. Bij het zinken van de laatste boot verschijnt er een overwinningsboodschap.
  - Je moet per speler een lijst kunnen geven van de reeds geprobeerde schoten en of het raak was.
  - Je moet per speler een overzicht kunnen geven van zijn naam en alle boten met hun naam en hun toestand. Dus clear, sunk met coördinaten of een lijst van (eventueel) geraakte compartimenten en een lijst van (eventueel) niet geraakte compartimenten.

    Speler1: Sam SS Wiekvorst: - Clear: (2,2),(2,3) SS Antwerpen: - Clear: (1,1), (1,2),(1,3) - Hit: (1,4) SS Retie: - Sunk: (3,3),(3,4)
- Je moet per speler een lijst bijhouden van schepen die hij heeft doen zinken en deze kunnen weergeven.

    Sunck: - SS Eindhoven (0,1),(0,2),(0,3) - SS Rotterdam (1,1),(1,2)

2. Om het basis programma uit te breiden gaan we extra functionaliteit toevoegen:
   - Je moet voor we beginnen met het spel de lengte en de breedte van het speelveld kunnen aanpassen.
   - Voeg een lijst van commandanten toe. Een commandant heeft de volgende eigenschappen:
     - Naam
     - Lijst met 1 of meerdere boten.

     Deze lijst van commandanten moet je kunnen afdrukken samen met de huidige van hun schepen.
   - Je moet per speler het schotenveld en het botenveld grafisch kunnen weergeven in twee aparte delen. Een compartiment van een boot is een B, een geraakt compartiment is een H en de zee een G. Voor het schotenveld gelijkaardig met B mis en H raak.

3. Om de overige punten te implementeren:
   - Schrijf het geheel door gebruik te maken van OO.
   - Geef de keuze om speler 1 en speler 2 te vervangen door een computer speler nadat de boten geplaatst zijn. Het volstaat deze speler ad random schoten te laten afvuren.
   - Mogelijkheid om het spel naar een bestand weg te schrijven en weer terug uit te lezen.

BELANGRIJK: Een schip is één entiteit en komt slecht één keer voor in de database. Dit wil dus zeggen dat als een schip moet voorkomen in verschillende lijsten, er niet voor elke lijst (speler, commandant) een nieuw schip wordt aangemaakt.

Veel succes

Academiejaar 2007 - 2008 - 2de zittijd

Afspraken

- Zet al de module bestanden die je gebruikt hebt in een archief met als naam “srolnummer.Arc”. Zoals bijvoorbeeld “s045480.Arc”.
- Gebruik enkel en alleen het Oberon lettertype voor je code (geen kleurtjes).
- Hou je strikt aan de Oberon Code Conventions.
- Je mag al de code die je ZELF hebt geschreven gebruiken.
- Je mag je cursus gebruiken.
- Elke vorm van communicatie is verboden.

Opdracht

Mogelijke weergave van een speelveld:

```
X | # | B | B | # | #
----------------------
# | # | G | # | # | #
----------------------
G | # | B | # | # | #
----------------------
# | # | B | # | # | B
----------------------
B | # | B | G | # | #
----------------------
# | # | # | # | # | #
```

Je implementeert een spel dat voldoet aan de volgende beschrijving. De bedoeling van het spel is op een 2D grid zoveel mogelijk items te verzamelen zonder in een val verstrikt te raken. Dit doe je door je mannetje horizontaal of verticaal over het speelveld te bewegen en zo de verschillende items te verzamelen. Elke speler mag om de beurt één plaats bewegen. Concreet maak je een systeem dat twee of meer spelers bevat. Vooraf kunnen het aantal items en het aantal vallen worden opgegeven. Deze dienen willekeurig over het speelveld verspreid te staan. De verschillende elementen op het speelveld mogen elkaar initieel niet overlappen. Een speler heeft per val 80% kans om deze te ontmantelen. Het is toegestaan het bord met een vast aantal elementen te configureren. Het spreekt echter voor zich dat een dynamische oplossing meer punten oplevert.

Alle vereiste functionaliteit staat in volgorde van belangrijkheid. Het spreekt voor zich dat je punten afhankelijk zijn van de hoeveelheid functionaliteit die geïmplementeerd en **getest** is. Zorg ervoor dat alle geïmplementeerde functionaliteit makkelijk te verifiëren is via de Tool file. Vergeet ook niet om duidelijk te specificeren welke functionaliteit geïmplementeerd is.

1. In het basisprogramma mag je er vanuit gaan dat het speelveld een vaste grootte (10x10) heeft en er slechts twee spelers zijn. Ook hoeven er geen controles te gebeuren bij het plaatsen van de elementen of het bewegen van de spelers. Het systeem moet de volgende functionaliteit bevatten:
    - Je moet spelers een naam kunnen geven en ze op het bord kunnen plaatsen, hierbij moet je zelf de coördinaten te specificeren.
    - Je moet een lijst items en een lijst vallen kunnen toevoegen aan het speelveld, ook hier moet je zelf de coördinaten specificeren.
    - De spelers moeten om de beurt kunnen bewegen over het speelveld. Wanneer een speler sterft speelt de overblijvende speler alleen verder. Wanneer alle items

verzameld zijn of er geen speler meer in leven is wint de speler met het meeste items.
    - Je moet de status van een speler kunnen weergeven. Deze bevat zijn naam, huidige locatie, een lijst van items met hun locatie en een lijst van ontmantelde vallen met hun locatie.
    - Je moet de status van het bord kunnen weergeven. Dit omvat de locatie van alle spelers, items en vallen die nog overblijven. Eenmaal een val is ontmanteld of een item is verzameld verdwijnt deze van het bord.
    - Je moet het speeldveld grafisch kunnen wwergeven, zowel met als zonder vallen. Spelers worden weergegeven door een X, vallen door een B, en items door een G. De lege vakjes worden weergegeven door een #.
2. Om het basis programma uit te breiden gaan we extra functionaliteit toevoegen:
    - Er dienen controles te zijn bij het plaatsen van spelers, items en vallen. Ook mag het niet mogelijk zijn dat een speler van het bord wandelt.
    - De kans dat een speler een val overleeft moet per val aangepast kunnen worden, met als minimum waarde 50%.
    - Je moet voor we beginnen met het spel de lengte en de breedte van het speelveld kunnen aanpassen.
    - Je moet meerdere spelers kunnen toevoegen aan het spel.
3. Om de overige punten te verdienen kan je nog de volgende extra mogelijkheden implementeren:
    - Schrijf het geheel door gebruik te maken van OO.
    - Je moet, na het opgeven van de spelers en het aantal item en vallen, het speelveld automatisch kunnen configureren. De locaties voor de spelers, items en vallen worden hierbij willekeurig gekozen.
    - Geef de keuze om de spelers te vervangen door de computer. Het volstaat deze spelers om de beurt in een willekeurige richting een stap te laten zetten. Dit kan je gebruiken om een automatisch testscenario te maken waarbij elke stap wordt afgedrukt.

BELANGRIJK: een item of een val is één entiteit en komt slechts één keer voor in de database. Dit wil dus zeggen dat als een speler er voor zorgt dat een item of val van het bord verdwijnt, en bij in de speler zijn lijst terechtkomt, er geen nieuw object wordt aangemaakt.

Veel Succes

Academiejaar 2005 - 2006 - 1ste zittijd

Afspraken

- Zet *al* de module bestanden die je gebruikt hebt in een archief met als naam van het archief "s*rolnummer*.Arc". Zoals bijvoorbeeld "s985109.Arc"
- Gebruik enkel en alleen het Oberon lettertype voor je code
- Hou je strikt aan de Oberon Code Conventions
- Noem je (hoofd)module Festival.Mod
- Zorg dat je code compileert! Pas dus op met half afgewerkte uitbereidingen.
- Vermeld duidelijk wat je hebt geimplementeerd

Opdracht

Een aantal vrienden ontmoeten elkaar op een openluchtfestival. Om niet van de dorst om te komen brengt iedereen geen of meer flessen fruitsap mee. Op elk moment kan elke persoon een, geen of meerdere flessen in zijn bezit hebben. Af en toe dinken de mensen van de fles die ze het minst lang in bezit hebben. Soms geven ze de fles die ze reeds het langs in hun bezit hebben

door aan iemand anders. Verder bevat elke fles slechts een beperkte inhoud, die wordt meegegeven wanneer een persoon aan het gezelschap wordt toegevoegd en die het aantal slokken meet als een integer waarde. Wanneer een fles leeg is wordt ze door de persoon die ze op dat moment vast heeft op de grond gegooid. Houdt ook van alle personen bij hoeveel slokken ze reeds geconsummeerd hebben.

Schrijf een Oberon module die de huidige toestand van het gezelschap bijhoudt. Volgende procedures moeten worden voorzien:

- Initialize : Initialiseer het programma of zet het programma terug in de beginstaat, er is nog niemand van het gezelschap op het festival aanwezig. Dit moet ook automatisch gebeuren bij het inladen van de module.
- Arrive : Iemand komt toe op het festival. Deze procedure neemt als parameters eerst de naam van de persoon, en dan de types fruitsap (string) die hij bijheeft tesamen met de inhoud van elke fles (integer). Het aantal flessen is variabel. (Ga ervan uit dat de laatste fles die is opgegeven reeds het langst in zijn bezit is (nodig voor de Drink en Pass procedures)).
- Drink: Iedereen drinkt een slok van de fles die hij het minst lang heeft. Wanneer hierdoor een fles leeg geraakt wordt deze op de grond gegooid. Dit wordt gemeld aan de gebruiker.
- Pass: Deze procedure neemt 2 persoonsnamen als parameter, en geeft aan dat de eerste persoon de fles die hij reeds het langste heeft doorgeeft aan de tweede.
- Part: Iemand verlaat het festival en neemt alle flessen die hij op dat moment heeft mee naar huis.
- Print: Geef een overzicht van wie er is en wie wat in zijn bezit heeft, tesamen met de inhoud van de flessen.

Opmerking: Personen hebben altijd verschillende namen (om identificatie voor Pass mogelijk te maken). Flessen kunnen mogelijk dezelfde naam hebben (e.g. `Festival.Arrive`

```
Pieter Appelsap 2 Appelsap 5~). Merk op dat dit de opgave niet gemakkelijker of moeilijker maakt (enkel de la
```

```
Festival.Arrive Wouter Appelsap 10 Bessensap 2 Citroensap 1~
> Wouter added...
Festival.Arrive Peter Passievruchtensap 3 Water 1 Wijn 1~
> Peter added...
Festival.Arrive Kurt~
> Kurt added...
Festival.Drink
Festival.Print
> De huidige aanwezigen zijn:
>   Kurt -- Huidige consumptie: 0 slokken -- Geen flessen in zijn bezit
>   Peter -- Huidige consumptie: 1 slok -- bezit:
>        Passievruchtensap -> 2 slokken
>        Water -> 1 slok
>        Wijn -> 1 slok
>   Wouter -- Huidige consumptie : 1 slok -- bezit:
>        Appelsap -> 9 slokken
>        Bessensap -> 2 slokken
>        Citroensap -> 1 slok
Festival.Drink
Festival.Pass Wouter Kurt
Festival.Drink
> Peters fles Passievruchtensap  is leeg...
> Kurts fles Citroensap is leeg...
Festival.Print
> De huidige aanwezigen zijn:
>   Kurt -- Huidige consumptie: 1 slok -- Geen flessen in zijn bezit
>   Peter -- Huidige consumptie: 3 slokken -- bezit:
>        Water -> 1 slok
>        Wijn -> 1 slok
```

```
>   Wouter -- Huidige consumptie : 3 slokken -- bezit:
>        Appelsap -> 7 slokken
>        Bessensap -> 2 slokken
Festival.Part Wouter
Festival.Arrive Gunther Druivensap 2~
Festival.Drink
> Peters fles Water is leeg
Festival.Print
> De huidige aanwezigen zijn:
>   Gunther -- Huidige consumptie: 1 slok -- bezit:
>        Druivensap -> 1 slok
>   Kurt -- Huidige consumptie: 1 slok -- Geen flessen in zijn bezit
>   Peter -- Huidige consumptie: 4 slokken -- bezit:
>        Wijn -> 1 slok
```

Deze basisimplementatie staat op ongeveer 12 punten. Extra uitbreidingen zijn:

- Zorg ervoor dat Print de aanwezige personen alfabetisch weergeeft. (2 punten)
- Zorg ervoor dat bij een print operatie de namen van de flessen die een persoon bezit alfabetisch worden geschikt. (3 punten)
- In plaats van hun flessen op de grond te smijten, zijn de aanwezigen beschaafder geworden, ze smijten hun flessen nu in de vuilbak in plaats van op de grond. Zorg ervoor dat de procedure Print ook de inhoud van de vuilbak weergeeft. (2 punten)
- Schrijf code die de huidige status van het gezelschap opslaat in een bestand en daarna terug van zo'n bestand inleest. (3 punten)
- Houd van elke gebruiker bij wat hij reeds heeft gedronken. De Print operatie wordt dan ook uitgebreid om per persoon weer te geven van welke flessen hij reeds heeft gedronken EN voor elk van die flessen hoeveel er op dat moment nog inzit (6 punten)

Academiejaar 2005 - 2006 - 2de zittijd

Afspraken

- Zet *al* de module bestanden die je gebruikt hebt in een archief met als naam van het archief "s*rolnummer*.Arc". Zoals bijvoorbeeld "s985109.Arc"
- Gebruik enkel en alleen het Oberon lettertype voor je code
- Hou je strikt aan de Oberon Code Conventions
- Noem je (hoofd)module CD.Mod en je toolfile CD.Tool
- Zorg dat je code compileert! Pas dus op met half afgewerkte uitbereidingen.
- Vermeld duidelijk wat je hebt geimplementeerd

Opdracht

Schrijf een stuk code dat je cd collectie beheert... Een CD bestaat uit een Groepsnaam, een Titel en een uniek Identificatienummer (ID). Elke CD bevat een lijst van tracks (nummers). Voor elke track moet de Titel en de Lengte (in seconden) van de track bijgehouden worden. Schrijf een Oberon module `CD.Mod`, die een database van je CD collectie bevat. Volgende methodes moeten geimplementeerd worden:

- `Add` : Deze methode voegt een CD toe aan de database, er wordt achtereenvolgens de Groepsnaam, de Titel en daarna de Track-gegevens opgegeven. De track-gegevens worden opgegeven als een opeenvolging van paren van strings en lengtes, e.g.

```
CD.Add "Therapy?" "Troublegum"
      "Knives" 116
      "Screamager" 156
      "Hellbelly" 200
```

```
     "Stop It You're Killing Me" 230
     "Nowhere" 146
     "Die Laughing" 168
     "Unbeliever" 208
     "Trigger Inside" 236
     "Lunacy Booth" 235
     "Isolation" 190
     "Turn" 229
     "Femtex" 194
     "Unrequited" 183
     "Brainsaw" 238 ~
```

Het ID wordt toegewezen door de database, het aantal tracks per cd is natuurlijk variabel.

- `DeleteByID` : Dit commando verwijdert een of meer cd's op ID, e.g.

```
CD.DeleteByID 6 9 12 ~
```

verwijderd CD's met ID 6, 9 en 12. Meldt als er ID's niet bestaan.

- `DeleteByBand` : Dit commando verwijdert *alle* cd's van een of meerdere gegeven groepen, e.g.

```
CD.DeleteByBand "Kreator" "Marillion" ~
```

- `List` : Geef een lijst van alle CD's, tesamen met hun ID.
- `Tracks` : Geef een lijst van alle tracks op een of meerdere CDs (gespecifieerd op ID), e.g.

```
CD.Tracks 9 12 3 1 ~
```

Deze basisimplementatie staat op ongeveer 12 punten. Extra uitbreidingen zijn:

- Schrijf code die de lijst met CD's en tracks naar een bestand schrijft en een weggeschreven bestand terug inleest. Het inlezen mag de huidige inhoud van de database volledig vervangen door de inhoud van het bestand. (schrijven 1 punt, lezen 3 punten)
- Schrijf een procedure `sort` die de albums op groep sorteert (2 punten).
- Schrijf code die een playlist beheert... (7 punten) De `play` methode neemt een lijst met CD ID's en tracknummers aan en plaatst deze in de playlist, de methode `show` toont de huidige playlist, tesamen met welke track nu aan het spelen is:

```
> CD.play <CDID_1> <tracknummer_1> <CDID_2> <tracknummer_2> ... ~
> CD.show ~
```

Een voorbeeld dialoog zou zijn

```
> CD.show ~
The playlist is empty
> CD.play 1 4 1 5 1 7~
Added "Stop It You're Killing Me" from album "Troublegum" (now playing)
Added "Nowhere" from album "Troublegum"
Added "Unbeliever" from album "Troublegum"
> CD.show
The current time is 12345
Now playing "Stop It You're Killing Me" by "Therapy?"
```

```
"Nowhere" by "Therapy?" (starting in 50 seconds)
"Unbeliever" by "Therapy?" (starting in 196 seconds)

(wacht 10 seconden)

> CD.show
The current time is 12355
Now playing "Stop It You're Killing Me" by "Therapy?"
"Nowhere" by "Therapy?" (starting in 40 seconds)
"Unbeliever" by "Therapy?" (starting in 186 seconds)

(wacht 45 seconden)

> CD.show
The current time is 12400
Now playing "Nowhere" by "Therapy?"
"Unbeliever" by "Therapy?" (starting in 141 seconds)
```

Gebruik hiervoor eventueel de methode TimeStamp die de tijd (in seconden sinds het begin van de dag) teruggeeft als een LONGINT waarde. Je mag dan veronderstellen dat alle commando's op dezelfde dag worden uitgevoerd om zo de wrap-around te vermijden.

```
IMPORT Utilities;

PROCEDURE TimeStamp() : LONGINT;
VAR
    d : Utilities.TDateTime;
BEGIN
    d := Utilities.Now();
    RETURN ( d.Second + 60 * ( d.Minute + 60 * d.Hour ) );
END TimeStamp;
```

Voorbeeld dialogen

```
CD.Add "Therapy" "Troublegum"
►►►►►►►►►"TT1" 10
►►►►►►►►►"TT2" 11
►►►►►►►►►"TT3" 20
►►►►►►►►►"TT4" 30
~
CD.Add "Therapy" "Semi-Detached"
►►►►►►►►►"TT5" 10
►►►►►►►►►"TT6" 5
~
CD.Add "Kreator" "Violent Revolution"
►►►►►►►►►"KT1" 10
►►►►►►►►►"KT2" 20
~
CD.Add "The Beatles" "Abbey Road"
►►►►►►►►►"BT1" 10
►►►►►►►►►"BT2" 20
►►►►►►►►►"BT3" 30
~
CD.DeleteByID 5 4 3 2~
CD.List
```

```
Added album Troublegum of band Therapy
Added TT1 ►: ►Length = 10
Added TT2 ►: ►Length = 11
Added TT3 ►: ►Length = 20
Added TT4 ►: ►Length = 30
Added album Semi-Detached of band Therapy
Added TT5 ►: ►Length = 10
```

```
Added TT6 ?: ?Length = 5
Added album Violent Revolution of band Kreator
Added KT1 ?: ?Length = 10
Added KT2 ?: ?Length = 20
Added album Abbey Road of band The Beatles
Added BT1 ?: ?Length = 10
Added BT2 ?: ?Length = 20
Added BT3 ?: ?Length = 30
CD with id 4 deleted
CD with id 3 deleted
CD with id 2 deleted
Album : <Troublegum> of band <Therapy> ?(1)
```

```
CD.Add "Therapy" "Troublegum"
??????????"TT1" 10
??????????"TT2" 11
??????????"TT3" 20
??????????"TT4" 30
~
CD.Add "Therapy" "Semi-Detached"
??????????"TT5" 10
??????????"TT6" 5
~
CD.Add "Kreator" "Violent Revolution"
??????????"KT1" 10
??????????"KT2" 20
~
CD.Add "The Beatles" "Abbey Road"
??????????"BT1" 10
??????????"BT2" 20
??????????"BT3" 30
~
CD.DeleteByBand Therapy "The Beatles"~
CD.List
```

```
Added album Troublegum of band Therapy
Added TT1 ?: ?Length = 10
Added TT2 ?: ?Length = 11
Added TT3 ?: ?Length = 20
Added TT4 ?: ?Length = 30
Added album Semi-Detached of band Therapy
Added TT5 ?: ?Length = 10
Added TT6 ?: ?Length = 5
Added album Violent Revolution of band Kreator
Added KT1 ?: ?Length = 10
Added KT2 ?: ?Length = 20
Added album Abbey Road of band The Beatles
Added BT1 ?: ?Length = 10
Added BT2 ?: ?Length = 20
Added BT3 ?: ?Length = 30
Album : <Violent Revolution> of band <Kreator> ?(3)
```

```
CD.List
CD.Add "Therapy" "Troublegum"
??????????"TT1" 10
??????????"TT2" 11
??????????"TT3" 20
??????????"TT4" 30
~
CD.Add "Therapy" "Semi-Detached"
??????????"TT5" 10
??????????"TT6" 5
~
CD.Add "Kreator" "Violent Revolution"
??????????"KT1" 10
??????????"KT2" 20
```

```
~
CD.Add "The Beatles" "Abbey Road"
��������"BT1" 10
��������"BT2" 20
��������"BT3" 30
~
CD.Play 1 3 ⍰ 2 3 ⍰ 2 2 ⍰ ⍰3 ⍰3 ⍰ ⍰ 3 2~
CD.Show
CD.Show
...
CD.Show
CD.Play 1 3 ⍰ 2 3 ⍰ 2 2 ⍰ ⍰3 ⍰3 ⍰ ⍰ 3 2~
CD.Show
CD.Show
...
```

```
No albums in database
Added album Troublegum of band Therapy
Added TT1 ⍰: ⍰Length = 10
Added TT2 ⍰: ⍰Length = 11
Added TT3 ⍰: ⍰Length = 20
Added TT4 ⍰: ⍰Length = 30
Added album Semi-Detached of band Therapy
Added TT5 ⍰: ⍰Length = 10
Added TT6 ⍰: ⍰Length = 5
Added album Violent Revolution of band Kreator
Added KT1 ⍰: ⍰Length = 10
Added KT2 ⍰: ⍰Length = 20
Added album Abbey Road of band The Beatles
Added BT1 ⍰: ⍰Length = 10
Added BT2 ⍰: ⍰Length = 20
Added BT3 ⍰: ⍰Length = 30
Track TT3 of album Troublegum now added (playing)
Track number 3 out of range for album Semi-Detached
Track TT6 of album Semi-Detached now added
Track number 3 out of range for album Violent Revolution
Track KT2 of album Violent Revolution now added
----------------------------> 59873
⍰ Now playing : TT3 (17 seconds remaining)
⍰ Queued : TT6 by Therapy (Starting in 17 seconds)
⍰ Queued : KT2 by Kreator (Starting in 22 seconds)
----------------------------> 59879
⍰ Now playing : TT3 (11 seconds remaining)
⍰ Queued : TT6 by Therapy (Starting in 11 seconds)
⍰ Queued : KT2 by Kreator (Starting in 16 seconds)
----------------------------> 59885
⍰ Now playing : TT3 (5 seconds remaining)
⍰ Queued : TT6 by Therapy (Starting in 5 seconds)
⍰ Queued : KT2 by Kreator (Starting in 10 seconds)
----------------------------> 59888
⍰ Now playing : TT3 (2 seconds remaining)
⍰ Queued : TT6 by Therapy (Starting in 2 seconds)
⍰ Queued : KT2 by Kreator (Starting in 7 seconds)
----------------------------> 59892
⍰ Now playing : TT6 (3 seconds remaining)
⍰ Queued : KT2 by Kreator (Starting in 3 seconds)
----------------------------> 59895
⍰ Now playing : KT2 (20 seconds remaining)
----------------------------> 59897
⍰ Now playing : KT2 (17 seconds remaining)
----------------------------> 59902
⍰ Now playing : KT2 (13 seconds remaining)
----------------------------> 59905
⍰ Now playing : KT2 (10 seconds remaining)
----------------------------> 59908
⍰ Now playing : KT2 (7 seconds remaining)
Track TT3 of album Troublegum now added
Track number 3 out of range for album Semi-Detached
Track TT6 of album Semi-Detached now added
Track number 3 out of range for album Violent Revolution
Track KT2 of album Violent Revolution now added
----------------------------> 59915
```

```
 Now playing : TT3 (20 seconds remaining)
 Queued : TT6 by Therapy (Starting in 20 seconds)
 Queued : KT2 by Kreator (Starting in 25 seconds)
----------------------------> 59921
 Now playing : TT3 (14 seconds remaining)
 Queued : TT6 by Therapy (Starting in 14 seconds)
 Queued : KT2 by Kreator (Starting in 19 seconds)
----------------------------> 59923
 Now playing : TT3 (12 seconds remaining)
 Queued : TT6 by Therapy (Starting in 12 seconds)
 Queued : KT2 by Kreator (Starting in 17 seconds)
----------------------------> 59927
 Now playing : TT3 (8 seconds remaining)
 Queued : TT6 by Therapy (Starting in 8 seconds)
 Queued : KT2 by Kreator (Starting in 13 seconds)
```

Academiejaar 2004 - 2005

Afspraken

- Zet *al* de module bestanden die je gebruikt hebt in een archief met als naam van het archief "s*rolnummer*.Arc". Zoals bijvoorbeeld "s045480.Arc".
- Gebruik enkel en alleen het Oberon lettertype voor je code.
- Hou je strikt aan de Oberon Code Conventions.

Bestand:Images/maggie.jpg

Opdracht

Maak een programma voor het plaatsen en vrijlaten van gevangenen. Gevangenen behoren tot één gevangenis. Een gevangenis heeft een unieke lokatie en een lijst van gevangenen. Van elke gevangene moeten volgende gegevens bijgehouden worden: een unieke code van `10` cijfers, de naam en voornaam (apart!), en het geslacht.

Als er lijsten moeten worden weergegeven, dan moet dit in tabelvorm! Zorg hierbij voor een propere weergave! Getallen worden rechts uitgelijnd, tekst links. Werk met het karakter `|` als scheiding tussen twee kolommen, en voorzie ook een hoofding. Voor een lijst van alle gevangenissen wordt dit:

```
 Lokatie          | #Gevangenen
------------------+-------------
 Antwerpen        |          24
 Dendermonde      |           7
 Gent             |          35
```

Implementeer telkens procedures om:

- een gevangenis toe te voegen (let op: de lokatie moet uniek zijn!);
- een lijst weer te geven van alle gevangenissen, alfabetisch gesorteerd op lokatie, met het aantal gevangenen erbij vermeld (zie voorbeeld);
- een veroordeelde als gevangene toe te voegen aan een gevangenis (let op: de code moet uniek zijn over alle gevangenissen heen!);
- een lijst weer te geven van alle gegevens van alle gevangenen van een bepaalde gevangenis op basis van de lokatie, alfabetisch gesorteerd op naam;
- een gevangene te kunnen vrijlaten (verwijderen) op basis van de code.

Academiejaar 2003 - 2004 - 1ste zittijd

Maak een Oberon implementatie van het spel *OXO*. Gegeven een speelbord van 4 op 4 tekent elke speler beurtelings een O of een X in één van de nog lege vakjes. Per OXO die hierdoor kan gevormd worden (in eender welke richting!), wordt er 1 punt toegevoegd aan de score van deze speler. Merk op dat het programma de vorming van OXO zelf moet detecteren! Een speler blijft aan beurt zolang hij scoort. Het spel eindigt wanneer het laatste vakje wordt opgebruikt. De speler met de hoogste score is dan de winnaar.

De implementatie moet aan een aantal basisvoorwaarden voldoen (die op 16 van de 20 punten staan). Daarnaast zijn er extra punten te verdienen met de implementatie van één of meer uitbreidingen naar keuze. Lees na de basisfunctionaliteit hieronder ook meteen de uitbreidingen alvorens te beginnen programmeren!

Basis

(op 16 van de 20 punten)

- Er moet op een speelbord van 4 op 4 vakjes gespeeld kunnen worden.
- Het aantal spelers is beperkt tot twee.
- Er moet een procedure bestaan om het spel ten allen tijde te starten met twee parameters: de naam van de eerste speler en de naam van de tweede speler.
- Er moet een procedure bestaan om de volgende beurt in te geven a.d.h.v. coördinaten en een aanduiding O of X. Het programma dient zelf te weten welke speler aan beurt is!
- Na elke (!) beurt moet het scherm leeg gemaakt worden en volgende informatie getoond worden:
  - Het nieuwe speelbord (met aanduiding van de coördinaten!).
  - Het scorebord.
  - Welke speler aan beurt is.

Een voorbeeld van de uitvoer (laatste beurt: X op $(1,1)$ door Sam):

```
  +---+---+---+---+
3 |   |   |   |   |
  +---+---+---+---+
2 | O |   | O |   |
  +---+---+---+---+
1 | O | X |   |   |
  +---+---+---+---+
0 | O |   | O | X |
  +---+---+---+---+
    0   1   2   3

Sam: 2
Frodo: 0

Speler aan beurt: Sam
```

Uitbreidingen

- (1 punten) Maak de grootte van het speelbord variabel (hoeft dus geen vierkant te zijn!).
- (2 punten) Maak het spel OO.
- (4 punten) Maak procedures om het spel op te slaan en terug in te lezen.
- (5 punten) Maak het aantal spelers variabel.

Afspraken

- Zet *al* de module bestanden die je gebruikt hebt in een archief met als naam van het archief "s*rolnummer*.Arc". Zoals bijvoorbeeld "s033450.Arc".
- Gebruik enkel en alleen het Oberon lettertype voor je code.
- Gebruik *geen* kleuren in je code of commentaar.
- Hou je strikt aan de Oberon Code Conventions.
- Gebruik duidelijke procedure namen.

Academiejaar 2003 - 2004 - 2de zittijd

Maak een programma voor het beheer van een kleinhandel. Het programma moet een lijst (van *onbeperkte* lengte!) bijhouden van de produkten die de kleinhandel aanbiedt. Per produkt moet de volgende informatie bijgehouden worden: een cijfercode, de naam van het produkt, de eenheidsprijs, en het aantal in voorraad.

De kleinhandel vult zijn voorraad aan door leveringen van een groothandel. Een groothandel moet **niet** geïmplementeerd worden! De leveringen van de groothandel aan de kleinhandel gebeuren in het programma gewoon via parameterdoorgave vanuit het Tool-bestand. Hierbij dient er per produkt achtereenvolgens opgegeven te worden: de cijfercode, de naam van het produkt, de eenheidsprijs en het aantal. Op dezelfde manier moeten er leveringen aan klanten kunnen gebeuren, maar hierbij dienen enkel de cijfercode en het aantal opgegeven te worden.

1. Zorg ervoor dat de lijst met produkten goed werkt (zowel toevoegen als verwijderen).

2. Zorg voor een propere (!) afdruk van alle produkten, in tabelvorm, gesorteerd op code (getallen worden rechts uitgelijnd, de rest links, prijzen met twee cijfers na de komma!). Bijvoorbeeld:

```
code | produkt                   | eh prijs | aantal
-----+---------------------------+----------+--------
  11 | Le Petit Prince           |    13.99 |      7
  37 | Solitudes - Forest Piano  |     7.95 |     10
 258 | Eureka 3D Puzzle          |     3.50 |     12
```

3. Voorzie een procedure voor leveringen van een groothandel. Een voorbeeld van de parameterlijst: `37 ``Solitudes - Forest Piano` *`6.95 25 129 ``For Love of the Game`* `9.95 20 258 ``Eureka 3D Puzzle` *`3.50 8`*. *Dit wil zeggen dat er* `25` *eenheden van produkt* `37`, `20` *van het produkt* `129`, *en* `8` *van produkt* `258` *werden geleverd. Merk op dat de cijfercode steeds uniek moet zijn, de produktnaam niet! Als de cijfercode reeds bestaat, moet de produktnaam en de prijs overschreven worden met de nieuwe gegevens indien ze niet overeenkomen. Druk na deze levering de tabel terug af. De nieuwe situatie in het voorbeeld is als volgt:*

```
code | produkt                   | eh prijs | aantal
-----+---------------------------+----------+--------
  11 | Le Petit Prince           |    13.99 |      7
  37 | Solitudes - Forest Piano  |     6.95 |     35
 129 | For Love of the Game      |     9.95 |     20
 258 | Eureka 3D Puzzle          |     3.50 |     20
```

4. Voorzie op dezelfde manier een procedure om zelf te leveren aan klanten. Hierbij moeten enkel de parameters cijfercode en aantal worden doorgegeven. Een voorbeeld van de parameterlijst: `11 1 258 2`. Dit wil zeggen dat er `1` eenheid van produkt `11`, en `2` van produkt `258` werd verkocht geleverd. Druk de tabel af met een overzicht van de verkochte produkten en de totale kostprijs, en nadien de globale tabel. In het voorbeeld:

```
 code | produkt                   | eh prijs | aantal
------+---------------------------+----------+--------
   11 | Le Petit Prince           |    13.99 |      1
  258 | Eureka 3D Puzzle          |     3.50 |      2
------+---------------------------+----------+--------
  TOT |                           |    20.99 |      3
```

```
 code | produkt                   | eh prijs | aantal
------+---------------------------+----------+--------
   11 | Le Petit Prince           |    13.99 |      6
   37 | Solitudes - Forest Piano  |     6.95 |     35
  129 | For Love of the Game      |     9.95 |     20
  258 | Eureka 3D Puzzle          |     3.50 |     18
```

Academiejaar 2002 - 2003 - 1ste zittijd

Dit examen bestaat uit twee oefeningen die *elk* gequoteerd zullen worden op 20 punten. De punten vermeld achter *basis* in oefening 2, en vóór elk puntje onder *uitbreidingen* van oefening 2 staan dus op 20. Het uiteindelijke cijfer voor dit examen wordt dan als volgt bepaald: het cijfer behaald op oefening 1 wordt herschaald naar 6, het cijfer behaald op oefening 2 wordt herschaald naar 14.

Opgave

1. Schrijf een programma dat van twee ingelezen gehele getallen recursief de grootst gemene deler berekent. Gebruik hiervoor het algoritme van *Euclides* ($m \geq n$):
   $gcd(m,n)=n$ als n deelt m
   $gcd(m,n)=gcd(n,m \bmod n)$ anders

2. Maak een Oberon implementatie van het spel *MasterMind*. Het spel gaat als volgt: er wordt een combinatie van k kleuren gekozen door het programma (uit acht verschillende kleuren, zijnde: wit, zwart, rood, blauw, oranje, geel, groen en bruin) en vervolgens moet de kleurencode binnen de p pogingen doorbroken worden. Na elke poging van de gebruiker wordt er *feedback* gegeven door het programma: ofwel staat de kleur op de juiste plaats (aanduiden met $+$); ofwel komt de kleur wel in de code voor, maar staat ze niet op de juiste plaats (aanduiden met $-$); ofwel komt de kleur helemaal niet in de code voor (geen aanduiding). De gebruiker van het programma moet dus via een aantal procedure oproepen zo snel mogelijk de code proberen te kraken. Lees na de basis hieronder ook de uitbreidingen voordat je begint te programmeren!

   #### Basis

   (12 punten)
   Voorzie, als basis voor het spel, volgende functionaliteiten en beperkingen:

   - De lengte k van de kleurencode mag je beperken tot 4.
   - Het maximum aantal pogingen p mag je beperken tot 10.
   - Zorg ervoor dat de kleurencombinatie *willekeurig* door de computer gekozen wordt (eenzelfde kleur mag meermaals in de code voorkomen!). (Hint: gebruik hiervoor de module RandomNumbers.)
   - Zorg ervoor dat bij elke nieuwe poging dezelfde functie wordt opgeroepen vanuit de Oberon werkomgeving, maar dan met andere parameters/kleuren (gebruik dus de

module In).

- Geef feedback na elke poging (zie boven).
- Zorg ervoor dat alle vorige pogingen zichtbaar (!) blijven.
- Druk na beëindiging van het spel (hetzij doordat het maximum aantal pogingen bereikt is, hetzij doordat de code doorbroken is) een passende boodschap af.
- Zorg ervoor dat op elk moment een nieuw spel kan begonnen worden.

Voorbeeld:

- Na de eerste poging:

```
=================================================
 1e poging | BL | ZW | OR | BR | - |   |   | +
```

- Na de tweede poging:

```
=================================================
 1e poging | BL | ZW | OR | BR | - |   |   | +
 2e poging | GE | BL | RO | BR |   | + |   | +
```

- Na de derde poging:

```
=================================================
 1e poging | BL | ZW | OR | BR | - |   |   | +
 2e poging | GE | BL | RO | BR |   | + |   | +
 3e poging | GR | BL | WI | BR | - | + | - | +
```

- Na de vierde poging:

```
=================================================
 1e poging | BL | ZW | OR | BR | - |   |   | +
 2e poging | GE | BL | RO | BR |   | + |   | +
 3e poging | GR | BL | WI | BR | - | + | - | +
 4e poging | WI | BL | GR | BR | + | + | + | +
-------------------------------------------------
Gewonnen!  Proficiat!
=================================================
```

Uitbreidingen

Als uitbreidingen op de basis voor het spel kan en mag je nog volgende functionaliteiten voorzien:

- (2 punten) Zorg ervoor dat de lengte k van de kleurencode kan worden opgegeven bij het starten van een nieuw spel.
- (2 punten) Zorg ervoor dat het maximum aantal pogingen p kan worden opgegeven bij het starten van een nieuw spel.
- (2 punten) Voorzie een *undo*-functie die je vanuit de Oberon werkomgeving kan aanroepen gedurende het spel. Hou dus een geschiedenis van de vorige pogingen bij.

- (3 punten) Bepaal de kleurencode bij het starten van het spel niet alleen puur random, maar laat de gebruiker de keuze tussen puur random en volgende methode: lees een volledig bestand in dat op elke nieuwe regel een andere kleurencode heeft staan en kies vervolgens willekeurig hieruit een kleurencode.

- (3 punten) Maak het spel object georiënteerd.

- (4 punten) Voorzie een functie om de huidige spelsituatie op te slaan in een bestand (zodat je op een later tijdstip het spel kan verder spelen) alsook een functie om een opgeslagen spelsituatie in te lezen van een bestand.

- (8 punten) Voorzie een functie waarmee je alle nog resterende mogelijkheden afdrukt, gegeven de feedback die je tot dan toe hebt. In het voorbeeld hierboven zou oproep van deze functie na de tweede poging dus het volgende moeten opleveren:

```
Mogelijkheid 1: GR BL GR BR
Mogelijkheid 2: GR BL WI BR
Mogelijkheid 3: WI BL GR BR
Mogelijkheid 4: WI BL WI BR
```

{Afspraken}

- Werk uitsluitend onder de Work directory van Oberon. Zet *al* de module bestanden die je gebruikt hebt in een archief met als naam “s*rolnummer*.Arc”. Zoals bijvoorbeeld “s024368.Arc”.
- Alle oefeningen dienen in Oberon geprogrammeerd te worden.
- Je krijgt drie uur de tijd, dus werk rustig en geconcentreerd.
- Alle zelf geschreven programma’s, voorbeeldprogramma’s, het boek en de cursus mogen gebruikt worden.
- Gebruik enkel en alleen het Oberon lettertype voor je code.
- Gebruik *geen* kleuren in je code of commentaar.
- Hou je strikt aan de Oberon Code Conventions op één punt na. Je hoeft namelijk *geen* commentaarblok te voorzien bij elke procedure, *wel* bovenaan elke module. Gebruik dus duidelijke procedure namen.

Academiejaar 2002 - 2003 - 2de zittijd

Schrijf een programma voor het beheer van de veestapel van nul of meer boeren. De veestapel kan bestaan uit nul of meer paarden, koeien, varkens of kippen. Hou voor elke boer een gelinkte lijst van dieren bij. Elke node in de gelinkte lijst mag slechts één dier voorstellen. De verschillende veestapels zelf moeten ook in een gelinkte lijst komen. Tussen de boeren onderling kan geruild worden waarbij volgende ruilvoorwaarden gelden:

- 1 paard kan geruild worden voor ofwel 2 koeien, ofwel 4 varkens, ofwel 48 kippen (of andersom).
- 1 koe kan geruild worden voor ofwel 2 varkens, ofwel 24 kippen (of omgekeerd).
- 1 varken kan geruild worden voor 12 kippen (of omgekeerd).

Het programma moet volgende operaties toelaten:

- Een boer moet kunnen worden toegevoegd.
- Een boer moet kunnen worden verwijderd.
- Paarden, koeien, varkens of kippen moeten kunnen worden toegevoegd aan de veestapel van een boer.

- De boeren moeten onderling dieren kunnen ruilen.

Afspraken

- Werk uitsluitend onder de Work directory van Oberon. Zet *al* de module bestanden die je gebruikt hebt in een archief met als naam "s*rolnummer*.Arc". Zoals bijvoorbeeld "s024368.Arc".
- Alle oefeningen dienen in Oberon geprogrammeerd te worden.
- Je krijgt drie uur de tijd, dus werk rustig en geconcentreerd.
- Alle zelf geschreven programma's, voorbeeldprogramma's, het boek en de cursus mogen gebruikt worden.
- Gebruik enkel en alleen het Oberon lettertype voor je code.
- Gebruik *geen* kleuren in je code of commentaar.
- Hou je strikt aan de Oberon Code Conventions op één punt na. Je hoeft namelijk *geen* commentaarblok te voorzien bij elke procedure, *wel* bovenaan elke module. Gebruik dus duidelijke procedure namen.

Academiejaar 2001 - 2002 - 1ste zittijd

1. Gebruik recursie om na te gaan of een ingelezen woord een palindroom is.

2. Schrijf een programma dat het spelletje *Nim* implementeert. Om Nim te spelen heb je twee spelers nodig en `15` lucifers die tussen beide spelers geplaatst worden zoals weergegeven in onderstaande figuur. Elke speler neemt om de beurt één, twee of drie lucifers. De speler die de laatste lucifer wegneemt verliest.

   Bestand:Images/nim.jpg

   Geef bij het begin van het spel *en* na elke beurt de nieuwe spelsituatie weer (gebruik hiervoor de module `OutExt`). Druk de lucifers dus af die nog overblijven zoals in het echte spel. Dus als een speler de middelste lucifer wegneemt dan moet er daar in uw uitvoer ook een lege plaats voorzien worden. Je kan de lucifers voorstellen door het karakter `|`. Verzorg de interactie met de gebruiker met commando-procedures gebruik makende van de module `In`. Vermeld ook telkens welke speler (A of B) aan de beurt is. Je hoeft de spelers geen naam te laten opgeven en je mag veronderstellen dat speler A steeds begint.

   Voorbeeld:

   - beginsituatie

   ```
   | | | | | | | | | | | | | | |
   0 1 2 3 4 5 6 7 8 9 10 11 12 13 14

   speler A is nu aan beurt
   ```

   - situatie na de eerste beurt (A heeft bijvoorbeeld lucifers 0, 2 en 13 weggenomen)

   ```
     |   | | | | | | | | | |   |
   0 1 2 3 4 5 6 7 8 9 10 11 12 13 14

   speler B is nu aan beurt
   ```

- situatie na de tweede beurt (B heeft bijvoorbeeld lucifer 8 weggenomen)

```
                    |     |  |  |  |  |     |  |  |  |     |
                  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14

                 speler A is nu aan beurt
```

- enz.

Voorzie ook de mogelijkheid om een nieuw spel te beginnen. Denk er hierbij aan dat het spel kan beëindigd worden voordat het effectief afgelopen is.
Voorzie ten slotte ook nog de mogelijkheid om terug te keren naar de vorige spelsituatie. Een *undo* functie dus. Maak hiervoor gebruik van gelinkte lijsten.

Enkele praktische afspraken:

- Werk uitsluitend onder de `Work` directory van Oberon. Zet *al* uw module bestanden in een archief met als naam “s*rolnummer*.Arc”. Zoals bijvoorbeeld “s015329.Arc”.
- Alle oefeningen dienen in Oberon geprogrammeerd te worden.
- Je krijgt drie uur de tijd, dus werk rustig en geconcentreerd.
- Alle zelf geschreven programma’s, de cursus en het boek mogen gebruikt worden.
- De kwaliteit van je code is veel belangrijker dan de kwantiteit!
- Gebruik enkel en alleen het Oberon lettertype voor je code.
- Gebruik *geen* kleuren in je code of commentaar.
- Hou je strikt aan de Oberon Code Conventions op één punt na. Je hoeft namelijk *geen* commentaarblok te voorzien bij elke procedure, *wel* bovenaan elke module. Gebruik dus duidelijke procedure namen.

Academiejaar 2001 - 2002 - 2de zittijd

1. De stelling van Pythagoras luidt: *In een rechthoekige driehoek is het kwadraat van de hypothenusa (*c*) gelijk aan de som van de kwadraten van de rechthoekszijden (*a *en* b*), of nog:* $a^2+b^2=c^2$*. Drietallen* (a,b,c) *die hieraan voldoen worden Pythagorische drietallen genoemd.*

   Schrijf nu een programma dat alle Pythagorische drietallen afdrukt waarbij zowel a, b als c variëren van 1 tot en met 100.

2. 1. Schrijf een procedure die nagaat of twee ingelezen strings anagrammen zijn. Twee strings zijn anagrammen indien de ene string een permutatie is van de andere. Of equivalent, indien beide strings uit exact dezelfde letters bestaan. Bijvoorbeeld, *elvis* en *lives*. De strings kunnen ook niet-alfabetische karakters bevatten, deze dienen genegeerd te worden. Er wordt ook geen onderscheid gemaakt tussen hoofdletters en kleine letters. Nog enkele voorbeelden zijn:

      - Year Two Thousand = A Year to Shut Down
      - George W. Bush = He Grew Bogus
      - A Decimal Point = I’m a Dot in Place
      - Get a Grip = Great Pig

   2. Onder de `Work` directory van Oberon staat een tekstbestand `anagrams.txt` dat een

verzameling strings bevat in willekeurige volgorde. Elke nieuwe string staat op een nieuwe regel. Lees nu dit bestand volledig in en geef weer welke strings anagrammen zijn van elkaar. Let op: het is mogelijk dat meer dan twee strings anagrammen zijn van elkaar!

Enkele praktische afspraken:

- Werk uitsluitend onder de `Work` directory van Oberon. Zet *al* uw module bestanden in een archief met als naam “s*rolnummer*.Arc”. Zoals bijvoorbeeld “s015329.Arc”.
- Alle oefeningen dienen in Oberon geprogrammeerd te worden.
- Alle zelf geschreven programma’s, de cursus en het boek mogen gebruikt worden.
- De kwaliteit van je code is veel belangrijker dan de kwantiteit!
- Gebruik enkel en alleen het Oberon lettertype voor je code.
- Gebruik *geen* kleuren in je code of commentaar.
- Hou je strikt aan de Oberon Code Conventions op één punt na. Je hoeft namelijk *geen* commentaarblok te voorzien bij elke procedure, *wel* bovenaan elke module. Gebruik dus duidelijke procedure namen.

Ontvangen van "http://www.winak.be/tuyaux/index.php?title=Inleiding_Programmeren"
Categorieën: Informatica | 1BINF

Inleiding tot C++

Uit Encyclopedia Academia

Bespreking

Inleiding tot C++	
Professor	Frans Arickx
Richting	Informatica
Jaar	1BINF

Dit vak sluit nu aan bij Inleiding Programmeren.

Vroeger was dit vak een onderdeel van het vak Software Engineering. Je zal enkele inleidingslessen in C++ krijgen. Sinds academiejaar 2008 - 2009 is dit vak gesplitst.

De leerstof van het vak is dezelfde als het eerste deel van de cursus Gevorderd Programmeren in de Tweede Bachelor Informatica, aangevuld met enkele vergelijkingen met Oberon-2. De evaluatie van het vak bestaat uit een kleinschalig C++ project en een theorie examen. Het theorie examen bestaat uit een schriftelijk deel zonder compiler, en een schriftelijk deel met compiler in de computerklas. Tijdens het tweede deel moeten er kleine stukjes code geschreven worden waarbij de student gebruik mag maken van de compiler om te verifiëren.

Alle vragen van voor het academiejaar 2008 - 2009 zijn de vragen die studenten kregen bij het vak Software Engineering.

Puntenverdeling

3/20 voor Jaaropdrachten, 7/20 voor het "eindwerkje" & 10/20 voor het schriftelijk deel.

Examenvragen

Academiejaar 2009 - 2010 - 1ste zittijd

Theorie

1. Het theorie examen voor het vak Inleiding tot C++ bestond uit vijf korte stukjes code met telkens de vraag of deze code compileerde of niet, met bijhorende uitleg. De voorbeelden waren meestal situaties die niet vaak voorkomen tijdens het programmeren en die soms wel logisch zijn maar toch een gevoel van twijfel durven oproepen. De essentie van elke oefening was het volgende:
    1. Overloading en conversion.
    2. Scope.
    3. Overloading en scope.
    4. Returnvalue is een pointer naar een variabele die op de stack staat.
    5. Prefix & postfix operatoren
2. Vervolgens volgende open essay-vragen:
    1. Bespreek het begrip conversie en vergelijk met Oberon-2.
    2. Bespreek het begrip overloading en hoe de compiler resolveert naar de meest geschikte kandidaat en bespreek het verschil met polymorfisme.
3. Uiteindelijk het gedeelte van het theorie examen waarbij men een compiler mag gebruiker (in de computerklas):
    1. Geef de declaratie van volgende types:

    1. Een reference naar een pointer naar een char.
    2. Een reference naar een functie met als argument een pointer naar een integer en als returnwaarde een pointer naar een integer.
    3. Een constante pointer naar een integer.
    4. Een array van 5 pointers naar een functie met als argument een pointer naar een char en als returnwaarde een reference naar een integer.
    5. Een pointer naar een constante char.
    6. Een functie met als argument een integer en als returnwaarde een pointer naar een functie met als argument een pointer naar een constante double en als returnwaarde een float.
2. Schrijf een stukje code dat door een lijst van floats enumerate.
3. Schrijf een stukje code dat het verschil tussen het grootste en kleinste getal weergeeft van een array van getallen.
4. Schrijf een functie waar je een dynamische array van integers definieert, deze initialiseert en de pointer teruggeeft.

Academiejaar 2007 - 2008 - 1ste zittijd

Theorie

1. We hebben in C++ mogelijkheden gezien vergelijkbaar met import/export in Oberon. Bespreek deze zo volledig mogelijk.
2. In C++ bestaat zoiets als function overloading. Wat is het? Wat is het verschil met polymorfisme?

Ontvangen van "http://www.winak.be/tuyaux/index.php?title=Inleiding_tot_C%2B%2B"
Categorieën: Informatica | 1BINF