# Inleiding Programmeren (Oberon)



## Inleiding Programmeren (Oberon)

| | |
|---|---|
| Richting | Informatica |
| Jaar | 1BINF |

## Bespreking

**Sinds 2011 wordt dit vak in de programmeertaal Python gegeven.**

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 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 2011 - 2012 - 2de zittijd

1. Waarvoor wordt import gebruikt in een class? Leg uit en als je 1 van de volgende woorden gebruikt, leg dan ook uit: *function*, *module*, *package*.
2. Kan je elk recursief programma omzetten naar een equivalent programma zonder recursies?
3. Leg volgende programmacode uit:

```
def function(v):
print(v)
v[0] = 42
print(v)
print(v[:-1])

answer = [0, 1, 2, 3]
function(answer)
print(answer)
```

1. Definieer en leg uit 'self' in een class, gebruik maximaal 10 lijnen.
2. Duid in volgende programmacode de fouten aan en verbeter:

```
def isPaly(x):

n = length(x)

while i in range(n):
if x[i] !== x[-i]
return false
else:
return True
```

## 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 uitvoeringsbesluitenuitvoeringsbesluiten (de semantische aanvullingen) hierbij in, en waar vindt men deze terug? Geef een paar voorbeelden.
2. Bespreek de eigenschappen van het booleanboolean 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(enkel voorvoor dede informatica)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 compatibiliteitseigenschappencompatibiliteitseigenschappen 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(enkel voorvoor dede informatica)informatica) De code maakt op de run-time stack gebruik van de base−pointerbase−pointer (BP)(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 kk kleuren gekozen door het programma (uit nn verschillende kleuren). De gebruiker van het programma moet zo snel mogelijk de code proberen te kraken, met een maximum van pp 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 pp is variabel, en wordt bij aanvang van het spel ingelezen (als parameter van het commando Start). Als p≤0p≤0 is het aantal pogingen onbeperkt.
- (*) De lengte kk 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 w1w1 en w2w2 in, en vervang in de tekst alle instanties van het woord w1w1 door w2w2.
- 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 laatste uitbreiding (zie verder) wordt moeilijker).

---

```
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 of length = 10
Added TT2 of length = 11
Added TT3 of length = 20
Added TT4 of length = 30
Added album Semi-Detached of band Therapy
Added TT5 of length = 10
Added TT6 of length = 5
Added album Violent Revolution of band Kreator
Added KT1 of length = 10
Added KT2 of length = 20
Added album Abbey Road of band The Beatles
Added BT1 of length = 10
Added BT2 of length = 20
Added BT3 of 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 1010 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 44 op 44 tekent elke speler beurtelings een OO of een XX in één van de nog lege vakjes. Per OXOOXO die hierdoor kan gevormd worden (in eender welke richting!), wordt er 11 punt toegevoegd aan de score van deze speler. Merk op dat het programma de vorming van OXOOXO 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 1616 van de 2020 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 1616 van de 2020 punten)

- Er moet op een speelbord van 44 op 44 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 OO of XX. 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

X

op (1,1)(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

- (11 punten) Maak de grootte van het speelbord variabel (hoeft dus geen vierkant te zijn!).
- (22 punten) Maak het spel OO.
- (44 punten) Maak procedures om het spel op te slaan en terug in te lezen.
- (55 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 2525 eenheden van produkt 3737, 2020 van het produkt 129129, en 88 van produkt 258258 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 11 eenheid van produkt 1111, en 22 van produkt 258258 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 2020 punten. De punten vermeld achter *basis* in oefening 22, en vóór elk puntje onder *uitbreidingen* van oefening 22 staan dus op 2020. Het uiteindelijke cijfer voor dit examen

wordt dan als volgt bepaald: het cijfer behaald op oefening 11 wordt herschaald naar 66, het cijfer behaald op oefening 22 wordt herschaald naar 1414.

Opgave

1. Schrijf een programma dat van twee ingelezen gehele getallen recursief de grootst gemene deler berekent. Gebruik hiervoor het algoritme van *Euclides* (m≥nm≥n):
   gcd(m,n)=ngcd(m,n)=n als nn deelt mm
   gcd(m,n)=gcd(n,m modulo n)gcd(m,n)=gcd(n,mmodulon) anders

2. Maak een Oberon implementatie van het spel *MasterMind*. Het spel gaat als volgt: er wordt een combinatie van kk 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 pp 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

   (1212 punten)
   Voorzie, als basis voor het spel, volgende functionaliteiten en beperkingen:

   - De lengte kk van de kleurencode mag je beperken tot 44.
   - Het maximum aantal pogingen pp mag je beperken tot 1010.
   - 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:

- (22 punten) Zorg ervoor dat de lengte kk van de kleurencode kan worden opgegeven bij het starten van een nieuw spel.
- (22 punten) Zorg ervoor dat het maximum aantal pogingen pp kan worden opgegeven bij het starten van een nieuw spel.
- (22 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.
- (33 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.
- (33 punten) Maak het spel object georiënteerd.
- (44 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.

    - (88 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 1515 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 (cc) gelijk aan de som van de kwadraten van de rechthoekszijden (aa en bb), of nog*

   *a2+b2=c2*

   *a2+b2=c2*

   *. Drietallen (a,b,c)(a,b,c) die hieraan voldoen worden Pythagorische drietallen genoemd.*
   Schrijf nu een programma dat alle Pythagorische drietallen afdrukt waarbij zowel aa, bb als cc variëren van 11 tot en met 100100.

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.