TECHNISCHE UNIVERSITEIT EINDHOVEN
Faculteit Wiskunde en Informatica

TU/e

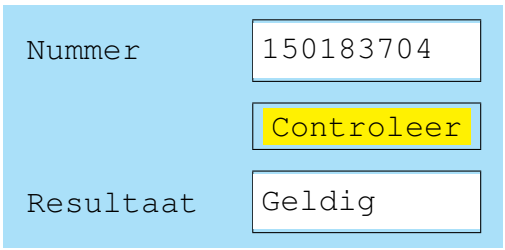
Tentamen 2IP05 (Programmeren) op donderdag 08 januari 2009, 09.00h–12.00h

---

Dit tentamen bestaat uit 2 vragen op 3 genummerde bladzijden. Bij het tentamen hoort het zip archief `2IP05-20090108.zip` met digitaal materiaal.

---

1. (Maximaal 6 punten) Maak een eenvoudige applicatie `ControleerNummers` met grafisch user interface (GUI) voor het controleren van bankrekeningnummers. Volg daarbij onderstaande aanwijzingen op. Zie ook figuur 1.



Figuur 1: Voorbeeld opmaak van het user interface (de kleuren zijn onbelangrijk)

   - De gebruiker kan een bankrekeningnummer van maximaal 10 cijfers invoeren in een `TEdit` veld, dat initieel leeg is. Langere invoer moet geweigerd worden (gebruik hiervoor eventueel de *property* `MaxLength`). Tevens moet het onmogelijk zijn om niet-numerieke invoer te geven. Gebruik het `OnChange` *event* om niet-numerieke invoertekens te verwijderen met behulp van de routine `DeCijfers` in de verderop beschreven **unit** `NummerControle`.
   - Het invoerveld is gelabeld via een `TLabel` met de tekst `Nummer`.
   - Onder het invoerveld staat een knop (`TButton`) met opschrift `Controleer`. De knop is alleen `Enabled` als het nummer uit 9 of 10 cijfers bestaat.
   - Daaronder staat een `ReadOnly` `TEdit` veld voor het resultaat van de controle, gelabeld (via `TLabel`) met de tekst `Resultaat`.
   - Het resultaatveld is aanvankelijk leeg en ook zodra de gebruiker het bankrekeningnummer wijzigt (denk aan het `OnChange` *event* van het invoerveld).
   - Direct na aanklikken van de knop verschijnt in het resultaatveld één van de twee teksten `Geldig` of `Ongeldig`, al naar gelang het ingevoerde bankrekeningnummer geldig is of niet.
   - De code voor de controle van een bankrekeningnummer staat in een aparte **unit** `NummerControle` met daarin de volgende functie specificaties:

function `DeCijfers` (**const** `AString`: **String**): **String**;
{ pre: True
ret: `AString` waaruit alle niet-cijfers verwijderd zijn }

function `BevatAlleenCijfers` (**const** `AString`: **String**): `Boolean`;
{ pre: True
ret: of `AString` uit alleen cijfers bestaat }

function `IsGeldigNummer` (**const** `ANummer`: **String**): `Boolean`;
{ pre: $n =$ `Length(ANummer)` ≤ 10 en
`BevatAlleenCijfers(ANummer)` en
`ANummer` $= c_n \ldots c_3 c_2 c_1$
ret: $(\sum_{w=1}^{n} w * c_w) \bmod 11 = 0$ }

Een raamwerk voor deze unit is gegeven; u implementeert `IsGeldigNummer`.

- `IsGeldigNummer` dient via een `Assert` de preconditie te controleren.
- Toelichting bij het bankrekeningnummer 150183704 in figuur 1: $1*4+2*0+3*7+4*3+5*8+6*1+7*0+8*5+9*1 = 132$ en $132 \bmod 11 = 0$, want $132 = 12*11$. Dit is dus inderdaad een geldig bankrekeningnummer.
- Lever het hele project in: `ControleerNummers.lpi`, `ControleerNummers.lpr`, `main.pas`, `main.lfm`, `nummercontrole.pas`.
- Denk aan het vermelden van naam, id.nr. en datum in de `*.pas` bestanden.

2. (Maximaal 4 punten) In een **unit** `PriorityQueue` zijn de volgende typedefinities opgenomen:

```
type
  PCell = ^ TCell; { pointer naar TCell }
  TCell = record
    FVal: Integer; { opgeslagen waarde }
    FNext: PCell; { nil of pointer naar volgende cel }
  end;
  { Voor p, q van type PCell schrijven we p → q wanneer p^.FNext = q }

  TIntList = PCell; { lijst van integer waarden }
    { extra data invariant: voor s van type TIntList geldt dat
      s in nil uitkomt via een eindig aantal → stappen (mogelijk nul stappen).
      Definieer list(s) = [] (lege lijst) als s = nil en anders
      list(s) = [ s^.FVal ] ++ list(s^.FNext) }

  TSortedIntList = TIntList; { gesorteerde lijst van integer waarden }
    { extra data invariant: voor s van type TSortedIntList geldt dat
      list(s) in oplopende volgorde gesorteerd is. }
```

Geef voor elk van onderstaande twee routine specificaties *twee* implementaties: één iteratief (met een lus), één recursief (zonder een lus). In totaal gaat het dus om vier routines.

- **function** `ListToStr` (`s`: `TIntList`; `sep`: **String**): **String**;
  pre: True
  ret: `list(s)` als string waarbij de eerste waarde is voorafgegaan door `sep` en elk van de overige waarden door één spatie.
  Converteer een `TIntList` naar een **String**.
  N.B. `ListToStr ( nil, 'x' ) = ''`, d.w.z. de lege string.
- **procedure** `InsertList` (**var** `s`: `TSortedIntList`; `v`: Integer);
  pre: True
  post: `list(s)` = `sorted(list(old s) ++ [ v ])`, waarbij `sorted(x)` = lijst van waarden in `x` oplopend gesorteerd
  Voeg een integer op de juiste plaats toe aan een `TSortedIntList`. Na afloop is de lijst nog steeds gesorteerd.

Schrijf uw implementaties in het aangeleverde bestand `priorityqueue.pas` en lever dit in.

Indien u een routine niet kunt implementeren, maak dan de body leeg, zodat het programma wel compileert.

Er is ook een Lazarus project `PrioQTestDriver` dat als handmatige test driver kan dienen. Dit project mag u wijzigen en levert u **niet** in.

Inleveren

Lever alle bestanden in één keer gezamenlijk in (zonder mapstructuur):

Opgave 1	**Lazarus**	**Delphi**
	`ControleerNummers.lpi`	
	`ControleerNummers.lpr`	`ControleerNummers.dpr`
	`main.pas`	`main.pas`
	`main.lfm`	`main.dfm`
	`nummercontrole.pas`	`nummercontrole.pas`

Opgave 2	**Lazarus**	**Delphi**
	`priorityqueue.pas`	`priorityqueue.pas`

(Einde van het tentamen.)