Turbo Pascal (deel 1)

MSX CLUB MAGAZINE 34

Erik van Bilsen

Scanned, ocr'ed and converted to PDF by HansO, 2001

Erik van Bilsen leert u het klappen van de Turbo Pascal zweep.

Turbo Pascal toepassen

Deze nieuwe serie behandelt de Turbo Pascal (versie 3.00 van Borland Inc.) aan de hand van enkele toepassingen zoals ik die heb gebruikt in onder andere het spel LETRIX. De serie is geschreven voor BASIC-programmeurs. Enige kennis van Turbo Pascal en machinetaal is wel handig, maar niet noodzakelijk.

Verschillen met basic

Het verschil tussen BASIC en Turbo Pascal (TP) is niet zo groot als velen denken. Veel BASIC-com-mando's zijn terug te vinden in TP en anders wel zonder veel moeite in TP te maken. Toch zijn er enkele belangrijke verschillen tussen de twee talen.

Compiler

Ten eerste werkt BASIC met een interpreter. Dat komt erop neer dat na het RUN-commando elke regel wordt vertaald om daarna pas te worden uitgevoerd. TP daarentegen werkt met een compiler. Het verschil met een interpreter is dat een compiler eerst het hele programma vertaald naar machinetaal, om vervolgens dat machinetaalprogramma te runnen. Hierdoor is een programma geschreven in TP vele malen sneller dan een programma dat is geschreven in BASIC.
[NvdR: Erik maakt hier de klassieke fout om te denken dat een interpreter een programma regel voor regel vertaalt in machinetaal en dan uitvoert. Dit is echt niet zo. De basic regel dient als data voor de interpreter die dan per instructie vertaalt of in de meeste gevallen de juiste BlOS-routine aanroept. We gunnen Erik het voordeel van de twijfel en menen dat hij de zaak niet nodeloos wilde compliceren.]

Lange namen

Een ander belangrijk verschil is dat in TP een lange, en dus overzichtelijke naamgeving aan variabelen is toegestaan. Zo zegt bijvoorbeeld de TP-regel:

Oppervlakte := Lengte * Breedte

veel meer dan de BASIC-equivalent

O = L * B.

Een voorwaarde is wel dat de gebruikte variabelen vooraf worden gedefinieerd.

Overzichtelijk

Verder is in TP de programmaop-bouw veel overzichtelijker. De aanroep van een BASIC-routine om bijvoorbeeld het geheugen van adres 40000 tot 41000 te vullen met waarde l zou er als volgt uit kunnen zien:

BA=40000: EA=41000: D=l: GOSUB 100

In regelnummer 100 wordt aan de hand van het beginadres (BA) en eindadres (EA) het geheugen gevuld met waarde D. Turbo Pascal kent geen regelnummers. In plaats daarvan heeft elke routine een naam. In TP ziet de bovenstaande regel er bijvoorbeeld als volgt uit:

VulGeheugen (40000,41000,1);

Hiermee is aan de bestaande commando's van TP het commando (procedure) VulGeheugen toegevoegd.

Basisprincipes

Om als BASIC programmeur snel aan de slag te kunnen met TP volgen hierna in het kort de belangrijkste principes om in TP te kunnen programmeren. Structuur
De opbouw van alle TP-program-ma's ziet er als volgt uit

```
PROGRAM [naam];

CONST
  [Declaratie van constanten]

TYPE
  [Declaratie van types]

VAR
  [Declaratie van variabelen]
[procedures/sub-programma's]

BEGIN
  [het hoofdprogramma]
END.
```

De afzonderlijke delen worden aan de hand van enkele voorbeelden behandeld. Het onderstaande voorbeeld kan in TP worden ingetypt door vanuit het TP-menu de E van Edit in te typen, gevolgd door de programmanaam. Je komt dan in de editor, een soort tekstverwerker, waarin het programma kan worden ingetypt. Om terug te keren naar het

menu (om bijvoorbeeld het programma te runnen) geef je de toetsen combinatie [CTRLJ-K, gevolgd door D.

```
PROGRAM Voorbeeld_l;

CONST
  PI = 3.1415927;

VAR
  Straal, Omtrek: REAL;

BEGIN
  Write ('Geef straal:');
  Readln (Straal);
  Omtrek := 2* PI* Straal;
  Writeln ('De omtrek is:', Omtrek);
END.
```

Constanten en variabelen

Het programma berekent de omtrek van een cirkel op basis van een door de gebruiker ingegeven straal. De eerste regel geeft de naam van het programma aan. Merk op dat elke regel of commando wordt afgesloten met een puntkomma (•).
Vervolgens wordt de constante PI gedeclareerd. De waarde van PI is in het programma niet te veranderen (vandaar de naam constante). Evenals de constanten moeten ook de gebruikte variabelen worden gedeclareerd. In BASIC is in principe elke variabele een reëel getal, tenzij het wordt gevolgd door een $ of % of anders is gedefinieerd met behulp van DEFSTR of DEF-INT. In TP moet vooraf van elke variabele worden aangegeven wat voor soort variabele het is.
De volgende soorten standaardvariabelen zijn te onderscheiden:

- INTEGER : Gehele getallen van-32768 t/m 32767
- REAL :Reële getallen
- BYTE : Deze getallen nemen l geheugenplaats (byte) in beslag en kunnen daarom een waarde hebben van O t/m 255
- CHAR: Een enkel karakter
- STRING[n]: een reeks karakters (string) van maximaal n tekens
- BOOLEAN: Waarde TRUE of FALSE, wordt later behandeld

```
PROGRAM Voorbeeld_l.l;

TYPE
  Str20 = STRING[20];

VAR
  Straal: REAL;

PROCEDURE Invoer (Tekst: str20; VAR Data: REAL);

BEGIN
  Write (Tekst);
  Readln (Data);
END;
```

```
FUNCTION Cirkelomtrek (Radius: REAL): REAL;

CONST
  PI = 3.1415927;

BEGIN
  Cirkelomtrek := 2 * PI * Radius;
END;

PROCEDURE Uitvoer (Tekst str20; Data: REAL);

BEGIN
  Writeln (Tekst, Data:9:4);
END;

BEGIN
  Invoer ('Geef straal:', Straal);
  Uitvoer ('De omtrek is:', Cirkelomtrek(Straal));
END.
```

Read en write

In het programma zijn de commando's Write en Read, al of niet gevolgd door ln, te vinden. De commando's zijn te vergelijken met de BASIC-commando's PRINT en INPUT. De toevoeging In geeft aan dat aan het einde van de opdracht naar de volgende regel moet worden gesprongen.
De pascal-regel

Writeln ('Test') ;

komt overeen met de BASIC-regel

PRINT "Test".
Evenzo komt de regel

Write ('Test') ;

overeen met de BASIC-regel

PRINT "Test"

Een verschil met BASIC is dat de strings worden aangeduid met een enkel aanhalingsteken (') in plaats van een dubbel ("). Verder staan alle parameters, zoals de tekst in het voorbeeld, altijd tussen haakjes.

Als laatste valt nog op dat in de formule een:- wordt gebruikt. Deze komt overeen met het BASIC-teken - en betekent in het voorbeeld: Aan de variabele Omtrek wordt de waarde van 2*PI*Straal toegekend. Pas op: in de constan-tendeclaratie wordt wel een enkele =

dus zonder: gebruikt.

Procedures en functies

In TP kunnen zoals gezegd zelf procedures en functies worden bijgemaakt (zoals het gebruik van subroutines in BASIC). Een -overdreven- voorbeeld (_1.1), waarin daarvan gebruikt wordt gemaakt is op de vorige pagina te vinden.

Dit programma geeft hetzelfde resultaat als het eerste voorbeeld, alleen is er nu gebruik gemaakt van twee procedures, Invoer en Uitvoer en een functie Cirkelomtrek.
Als eerste valt op dat er een type is gedefinieerd, namelijk str20. Dit type is van het soort STRING. Die toevoeging [20] geeft aan dat het gaat om strings van maximaal 20 tekens lang. Elke variabele, die wordt gedefinieerd als str20, is dus een string van maximaal 20 tekens.
Elke procedure of functie heeft dezelfde opbouw als een programma, met uitzondering van de puntkomma achter de END in plaats van de punt. De END aan het einde van de subroutine is gelijk aan de RETURN in BASIC.

Turbo Pascal

```
A := Abs(=2);
A := ArcTanfl);
A := Chr(65);
ClrScr;
A := Copy(Test ,2,2);
A := Cos(2);
Delay(tOO);
GotoXY(1,6);
A := Length(Test');
A:= Ord('A');
A := PosCes'.Test1);
A:= Random(3);
Randomize;
Read (A);
Readln (A);
A := Round(3.7);

A := Sin(2);
A := Sqr(3);
A:= Sqrt(9);
Str(3,A);
A := Trunc(3.7);

A := UpCase('a');

Val('3',A,Code);


Write (Test);
Writeln ('Test');
```

BASIC

```
A = ABS(=2)
A = ATN(1)
A$ = CHR$(65)
Cls
A$ = MID$(Test",2,2)
A = COS(2)
[wacht 100 microseconden]
LOCATEO,5
A = LENfTest")
A = ASC("A")
A = INSTR("es","Test")
A = INT(RND(1)*3)
A = RND(=TIME)
INPUT A;
INPUT A
A = INT(3.7 + 0.5)
[afronden, resultaat=4]
A = SIN(2)
A = 3A2     II
A = SQR(9) !!
A$ = STR$(3)
A = INT(3.7)
[afbreken, resultaat=3]
[omzetten naar hoofdletters,
resultaat='A']
A = VAL("3")
[Code is positie van eerste
niet om te zetten teken]
PRINT "Test";
PRINT Test"
```

```
10 ' *** Constante
20 PI = 3.1415927
30 ' *** Functie CirkelOmtrek
40 DEF FNC(R) = 2 * PI * R
50 '*** Hoofdprogramma
60 T$ = "Geef straal:": S = 0: GOSUB100: S = D
70 T$ = "De omtrek is: " : D = FNC(S) : GOSUB 200
80 END
90 ' *** Procedure Invoer
100 PRINT T$; : INPUT D: RETURN
190 ' *** Procedure Uitvoer
200 PRINT T$;D: RETURN
```

Parameters

Aan de procedure Invoer worden twee parameters doorgegeven, nl. Tekst en Data. In het hoofdprogramma wordt deze procedure aangeroepen. De parameters die daar worden doorgegeven zijn de constante 'Geef straal:' en de variabele 'Straal'. Dit zijn de formele parameters. In de procedure Invoer bevat de variabele Tekst vervolgens de string 'Geef straal:' en de variabele Data de waarde van de variabele Straal. Tekst en Data zijn de actuele parameters. In de aanhef van de procedure Invoer valt op dat het woordje VAR voor de variabele Data staat. Dit betekent dat als de variabele Data wordt veranderd, de doorgegeven parameter Straal mee wordt veranderd. Het voorafgaande staat schematisch bovenaan deze bladzijde weergegeven.
Nadat in de procedure de variabele Tekst op het beeldscherm wordt afgedrukt, wordt de gebruiker gevraagd de straal van de cirkel in te geven. Deze wordt vervolgens opgeslagen in de variabele Data. Bij terugkeer naar het hoofdprogramma krijgt de variabele Straal dan dezelfde waarde als Data (vanwege het woordje VAR). Deze variabele Straal wordt vervolgens doorgegeven aan de functie met de naam Cirkelomtrek.
Een functie wordt hetzelfde gedefinieerd als een procedure, met de uitzondering dat achter de definitie het type moet worden opgegeven van de waarde, die aan het hoofdprogramma teruggegeven wordt. Een uitleg van de gebruikte functie Cirkelomtrek verduidelijkt dit wellicht. Aan de functie Cirkel=omtrek wordt een parameter doorgegeven die vervolgens wordt opgeslagen in de variabele Radius van het type REAL. Het woordje REAL achter de functie geeft aan dat het resultaat van deze functie, hier de omtrek, als 'n reëel getal wordt doorgegeven aan het hoofdprogramma. Vervolgens kan de naam Cirkelomtrek worden gezien als een variabele van het type REAL, waaraan de uitkomst van de berekening moet worden toegekend. De functie kan als volgt worden aangeroepen:

A := Cirkelomtrek(5);

In het voorbeeld wordt de uitkomst van de functie in het hoofdprogramma meteen als parameter doorgegeven aan de procedure Uitvoer. In deze procedure valt tenslotte de toevoeging :9:4 achter Data op. Dit wil zeggen dat de waarde van de variabele Data (de omtrek) wordt weergegeven in totaal 9 posities, waarvan 4 cijfers achter de decimale punt.
Het gebruik van parameters is misschien wel het lastigste aspect van Turbo Pascal voor

programmeurs die alleen BASIC ervaring hebben. Parameters zijn echter heel nuttig en makkelijk te gebruiken. Ze stellen de programmeur in staat om eigen commando's en subroutines te maken. BASIC programmeurs raad ik aan het bovengenoemde TP-programma te vergelijken met de bovenstaande BASIC-versie. De vergelijking laat het nut van parameters zien en verduidelijkt de theorie. Ten behoeve van de overzichtelijkheid is ook in de basic versie het gebruik van subroutines overdreven. Het idee erachter is dat er een algemene invoer- en uitvoerroutine is ontworpen, die meerdere malen, voor verschillende toepassingen kan worden aangeroepen.

Praktische toepassingen

Met de basisprincipes van Turbo Pascal in een notedop wordt deze eerste aflevering besloten. Zoals elk begin bestond ook deze eerste aflevering uit de noodzakelijke theorie. Het feitelijke programmeren in Turbo Pascal wordt in de volgende afleveringen besproken aan de hand van praktische toepassingen, tips en trucs, zoals grafische en mu=ziekroutines en geheugenmanipulatie. Oefenen en creativiteit zijn daarvoor de enige sleutelwoorden om tot een resultaat zoals het programma LETRIX te komen.
Ter afsluiting een overzicht van veel voorkomende standaard procedures & functies in Turbo Pascal met hun BASIC equivalent op de vorige pagina en een overzicht van de TP editmogelijkheden hieronder.

Turbo Pascal Editor

Cursorverplaatsing

Naar begin woord links van cursor.....……...Ctrl-A
Naar begin woord rechts van cursor..... Ctrl-F
Scroll tekst 1 regel omlaag .....Ctrl-Z
Scroll tekst 1 regel omhoog .....Ctrl-W
Scroll tekst 1 pagina omlaag.....Ctrl-C
Scroll tekst 1 pagina omhoog .....Ctrl-R
Naar begin regel..... Ctrl-QS
Naar einde regel..... Ctrl-QD
Naar eerste regel scherm..... Ctrl-QE
Naar laatste regel scherm..... Ctrl-QX
Naar begin programmatekst..... Ctrt-QR
Naar einde programmatekst..... Ctrl-QC
Naar begin blok..... Ctrl-QB
Naar einde blok..... Ctrl-QP

Invoegen/verwijderen

Verwijder teken bij de cursor.....Ctrl-G
Verwijder teken links van de cursor.....Del
Verwijder hele regel.....Ctrl-Y
Verwijder tot einde regel.....Ctrl-QY
Verwijder woord rechts van cursor..... Ctrl-T
Voeg nieuwe regel in .....Ctrl-N
Wisselen invoegen/overschrijven..... Ctrt-V

Blokopdrachten

Markeer begin blok.....Ctrl-KB
Markeer einde blok.....Ctrl-KK
Markeer 1 woord.....Ctrl-KT
Kopieer blok..... Ctrl-KC
Verwijder blok.....Ctrl-KY
Verplaats blok.....Ctrl-KV
Lees blok van schijf..... Ctrl-KR
Schrijf blok naar schijf.....Ctrl-KW

Overig

Automatisch inspringen aan/uit.....Ctrl-QI
Verlaten editor (terug naar hoofdmenu)...Ctrl-KD

Zoeken en vervangen..... Ctrl-QA
Zoek string..... Ctrl-QF
Zoek volgend..... Ctrl-L
Herstel regel.....Ctrl-QL