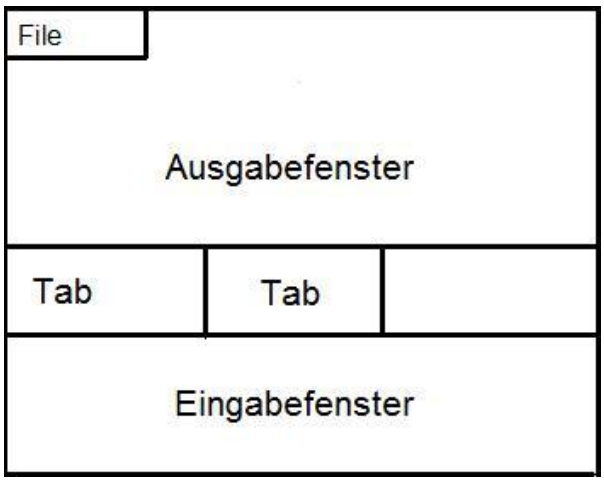
-bisher interaktive Anwendung



à Ziel: Erweitere die Funktionalität um selbst erzeugte Zusatzfunktionen

1. Einige Programmentwicklungserklärungsversuche

a) Man kann neue Eingabedateien öffnen mit Menü „File“, „New“. Es erscheint im Eingabebereich ein zweites „Tab“.

b) In der neuen Datei kann man die eigenen Funktionen eintippen. Wenn man fertig ist, kann man die neue Funktion mit „Strg/Ctrl“ + „Enter“ absenden. Dann prüft CoCoA die Syntax und wenn die Funktion ohne Fehler akzeptiert wird, ist sie einsatzbereit. **Austesten im interaktiven Fenster.**

c) Zum Schluss kann man den Inhalt des Dateifensters mit „File“ und „Save As…“ speichern. Als Dateiname ist *.coc üblich.

2. Define… End Define

Neue CoCoA-Funktionen werden erzeugt mit Define MeineFunktion(A, I)

A,I = Funktionsargumente

è kein „ ; “ , da der Befehl erst mit „End Define“ zu Ende ist

```
B : = A ^I;
Return B;
End Define;
A:=Mat[ [0,1],[0,0] ];
MeineFunktion(A,2);
```

à Befehl „Return“ zur Ausgabe des Ergebnisses

à Ergebnis : die Nullmatrix

3. Programmablaufsteuerung

a) Schleifenbefehl

```
For I:=1 To 10 Do
<Befehle>
End For;
```

Die Befehle werden 10 Mal ausgeführt. Dabei hat die Variable I die Werte 1, 2,…,10

b) Schleifenbefehl für Listen

è Sei L eine Liste.

Die Befehle werden für jedes Element der Liste 1 Mal abgearbeitet. Die Variable T enthält jeweils das momentane Element von L.

ForEach T **In** L **Do**

<Befehle>

End ForEach;

c) Schleifen mit unbestimmter Wiederholungszahl

While <logische Bedingung> Do
<Befehle>
End While;

Eine logische Bedingung ist eine Variable/Berechnung, die **TRUE oder FALSE** ergibt, z.B. N = 5 oder N < > 5 (N ungleich 5 in CoCoA).

Die Befehle werden immer wieder abgearbeitet, solange bis die logische Bedingung den Wert **FALSE** ergibt.

d) Verzweigungsbefehl

Die Befehle werden nur ausgeführt, wenn die logische Bedingung TRUE ergibt.

If <logische Bedingung> **Then**

<Befehle>

EndIf;

e) Doppelter Verzweigungsbefehl

Die Befehle 1 werden ausgeführt, wenn die logische Bedingung TRUE ergibt. Wenn sie FALSE ergibt, werden Befehle 2 ausgeführt.

If <logische Bedingung> **Then**
<Befehle 1>
Else
<Befehle 2>
EndIf;

Beispiel: Der euklidische Algorithmus

```
Define Euklid(A,B)
    If(A=0 And B=0) Then
    Return 0;
    EndIf;
        If A=0 Then
    Return AbsValue(B);          à Betrag von B
        EndIf;
        If B=0 Then
    Return AbsValue(A);          à Betrag von A
        EndIf;
    A:=AbsValue(A);
    B:=AbsValue(B);
```

```
    If A<B Then
                C:=A;
                A:=B;
                B:=C;
                EndIf;
        Rest:=1;
    While Rest <>0 Do
        Rest:=Mod(A,B);
        A:=B;
        B:=Rest;
    EndWhile;
        Return A;
End Define;
    Define AbsValue(B)
                If B<0 Then
        Return –B;
    Else Return B;
                EndIf;
    End Define;
```

C muss **nicht vorher** definiert werden. CoCoA macht dies selbst!

4. Die Hinterlistigen Listen (Listenkonstruktor)

Sei z.B. eine Liste, etwa L= 1..100; ß [1,2,3,4,5..,99,100]

a) G:=[N In L | Mod(N,2)=0];
   è G=[2,4,6,8…98,100]
b) H:=[N^2| N In L];
   è H=[1,4,..,10000];
c) I :=[N^2 | N In L And N<=10];
   è I:=[1,4,9..,100]

Beispiel: Angenommen es gibt eine Funktion

CharPoly(A), die das charakteristische Polynom einer Matrix A berechnet. Der Grundring sei K[x].

```
Define Eigenwerte(A);
        F:=CharPoly(A);
        L:=Factor(F);
        M:=[P In L | Deg(P[1]) = 1];
N:=[P[1] | P In M];
LL:=[CoeffOfTerm(1,F)/LC(F) | F In N];
        Return LL;
End Define;
```

Factor(x^3-1);

- è [[x^2+x+1,1],[x-1,1]]
- è Nullstelle ist 1.

ax –b könnte dort stehen x= b/a

CoeffOfTerm(T,F)

- è Koeffizient des Terms T im Polynom F