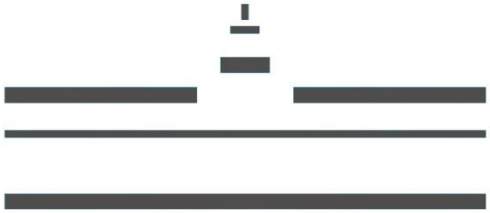


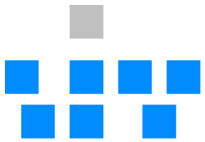
Westfälische
Wilhelms-Universität
Münster

› PL/I und Internet

München 2009

Eberhard Sturm
ZIV (Uni-RZ)
Münster
sturm@uni-muenster.de

wissen.leben
WWU Münster



ZENTRUM FÜR
INFORMATIONS
VERARBEITUNG

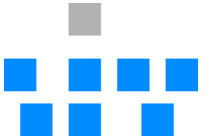
› Seit Jahren

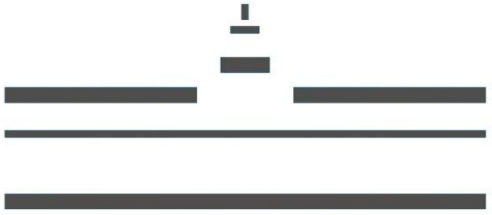
20:42

SHARE, G.U.I.D.E.

- Inheritance considered harmful
  SHARE-Europe Anniversary Meeting 1989 Amsterdam
- "Programmieren in PL/I", Vieweg, 1990
- Hi-GKS – Concepts for a New Graphic Standard Implemented using PL/I
  SHARE-Europe Anniversary Meeting 1991 Amsterdam
- Power vs. Adventure – PL/I and C
  G.U.I.D.E & SHARE Europe Joint Meeting Oktober 1994 in Wien

- Im Dschungel von Neu-C-Land: PL/I-Programmierung unter OS/2
  GUIDE-PL/I-Tagung November 1997 in Hamburg
- Schnittstellen zur Welt (PL/I und C, Unicode, REXX, Java, CGI, XML, OpenGL)
  GUIDE-ADL-Tagung November 2002 in Karlsruhe
- Mit XML zum PL/I-GUI
  GUIDE-ADL-Tagung April 2004 in Dortmund





Westfälische
Wilhelms-Universität
Münster

XML QueryPath area DOM macro jQuery offset

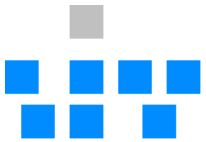
› PL/I und Bundesliga


München 2009

allocate() pliq package based HTML plifree() plisaxb refer Java parse Objekt JNI memconvert() do_foreach SAX

Eberhard Sturm
ZIV (Uni-RZ)
Münster
sturm@uni-muenster.de

wissen.leben
WWU Münster



ZENTRUM FÜR
INFORMATIONS
VERARBEITUNG

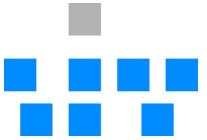
› Auf Knopfdruck

20:42

12. Spieltag (Samstag)

D:\Projekte\München\zzz_test_pliq.exe

```
Start ...

Webseite "http://www.kicker.de/news/fussball/bundesliga/spieltag/tabelle/liga/1/
tabelle/1/saison/2009-10/spieltag/1/inde
x.html" wurde geladen.
Dokument wurde in ein DOM ³bernommen.
Dokument wurde in Datei gespeichert.
pliq-Objekt wurde erzeugt.
DOM wird jetzt durchst÷bert!

Bayer Leverkusen      -  Eintracht Frankfurt  4:0 (3:0)
Bayern München        -  FC Schalke 04        1:1 (1:1)
1899 Hoffenheim       -  VfL Wolfsburg        1:2 (1:0)
VfL Bochum            -  SC Freiburg          1:2 (0:1)
Borussia M'gladbach   -  VfB Stuttgart       0:0 (0:0)
1. FSV Mainz 05       -  1. FC Nürnberg       1:0 (1:0)
Hannover 96           -  Hamburger SV         -:- (-:-)
Hertha BSC            -  1. FC Köln           -:- (-:-)
Werder Bremen         -  Borussia Dortmund    -:- (-:-)

Ok?
_
```



› Woher bekommt man die Ergebnisse?

20:42

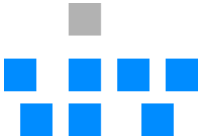
Internet

- Aus dem Internetauftritt der Zeitschrift **kicker**:

```
dcl Seite char (*) value
    ('http://www.kicker.de/news/fussball/bundesliga/'
    || 'spieltag/tabelle/liga/1/tabelle/1/saison/'
    || '2009-10/spieltag/1/index.html');
```

- Das sieht z. B. so aus:

```
<tr class="fest alt"> <td class="first">Sa</td> <td>24.10. 15:30</td> <td ><div
id="ctl00_PlaceHolderContent_ctl01_ctl00_repBegegnungen_ctl03_ctl02_showMe"> <div
id="ctl00_PlaceHolderContent_ctl01_ctl00_repBegegnungen_ctl03_ctl02_verlinkt"> <a
href="/news/fussball/bundesliga/vereine/1-bundesliga/bayern-muenchen-
14/vereinsinformationen.html"
id="ctl00_PlaceHolderContent_ctl01_ctl00_repBegegnungen_ctl03_ctl02_NameA" class="link"
Style="">Bayern München</a> </div> </div> </td> <td> - </td> <td ><div
id="ctl00_PlaceHolderContent_ctl01_ctl00_repBegegnungen_ctl03_ctl03_showMe"> <div
id="ctl00_PlaceHolderContent_ctl01_ctl00_repBegegnungen_ctl03_ctl03_verlinkt"> <a
href="/news/fussball/bundesliga/vereine/1-bundesliga/eintracht-frankfurt-
32/vereinsinformationen.html"
id="ctl00_PlaceHolderContent_ctl01_ctl00_repBegegnungen_ctl03_ctl03_NameA" class="link"
Style="">Eintracht Frankfurt</a> </div> </div> </td> <td class="alignright">2:1 (0:0)</td>
<td> </td> <td><a class="link" href="/news/fussball/bundesliga/spieltag/1-bundesliga/2009-
10/10/938111/spielanalyse_bayern-muenchen-14_eintracht-frankfurt-32.html">Analyse</a></td> <td
class="aligncenter" ><a href="/news/fussball/bundesliga/spieltag/1-bundesliga/2009-
10/10/938111/spielinfo_bayern-muenchen-14_eintracht-frankfurt-32.html"><img
src="http://mediadb.kicker.de/library/images/icon_info.png" width="12" height="13" alt=""
/></a></td> <td class="aligncenter last"></td> </tr>
```



› Und wie in PL/I?

20:42

Nur PL/I!

- 1. Versuch – plisaxb:

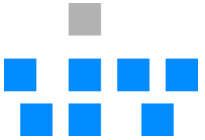
```
call plisaxb (E, P, 'file://meine_datei.xml');

call plisaxb (E, P,
'http://www.kicker.de/news/fussball/bundesliga/...');
```

=› Fehlschlag !!!

Und das ist auch gut so!

- 2. Versuch – JNI:

Java Native Interface



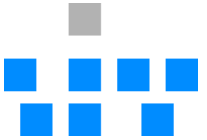
› Und wie doch in PL/I?

20:42

Java Native Interface

- C-Unterprogramm-Aufrufe von PL/I aus:
  1. Start der Java-Virtual-Machine
  2. Festlegen von Klassen
  3. Erzeugen von Objekten
  4. Festlegen von Methoden
  5. Aufruf von Methoden
  6. Stoppen der Java-Virtual-Machine

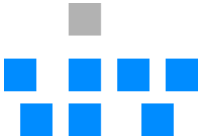
- Probleme:
  - unterschiedliche Datentypen
  - unübersichtliche Programmaufrufe



› Unser Hauptprogramm

Fragment 1

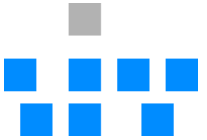
```
Programm: package options (reorder);
%include pliload;
dcl Seite char (*) value
    ('http://www.kicker.de/news/fussball/bundesliga/'
    || 'spieltag/tabelle/liga/1/tabelle/1/saison/'
    || '2009-10/spieltag/1/index.html');

Haupt: procedure (Parm) options (main noexecops);

dcl Parm       char (*) var parm nonasgn;
dcl DOK_länge fixed bin (31);
dcl DOK_ptr    ptr;
...
/* HTML-Dokument in ein Objekt laden: */
call pliloadhtml(DOK_ptr, DOK_länge, Seite);
...
call plifree (DOK_ptr);
end Haupt;
end Programm;
```



› JNI an sich

pliloadhtml

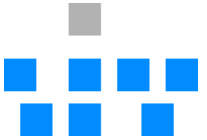
```
...
%include java;
dcl Vm_args type JavaVMInitArgs;
dcl Options dim (dim(P)) type JavaVMOption;
dcl P dim (1) ptr static
      init to (varz)('-Djava.compiler=NONE');

do I = 1 to dim(P);
   Options(I).OptionStringPtr = P(I);
   Options(I).extraInfoPtr = null();
   end;
Vm_args.version = '00010004'xn;
Vm_args.nOptions = dim(Options);
Vm_args.optionsPtr = addr(Options);
Vm_args.ignoreUnrecognized = JNI_FALSE;

I = JNI_CreateJavaVM(JavaVMPtr, JNIEnvPtr, Vm_args);
if I ¬= 0 then signal condition (Java_exception);
...
```



› Wie macht man's also?

pliloadhtml

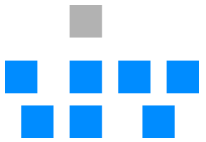
```
pliloadhtml:
procedure (Puffer_ptr, Pegelstand, Parmadresse);

%include java;
dcl Puffergröße fixed bin (31) value (200_000);
dcl Puffer_ptr  ptr            parm asgn;
dcl Pegelstand  fixed bin (31) parm asgn;
dcl Parmadresse char (*)       parm nonasgn;

Puffer_ptr = allocate(Puffergröße);
on condition (Java_exception) begin;
   java_handle_exception describe clear;
   Pegelstand = 0;
   goto Return;
   end;

java_start options ('-Djava.compiler=NONE');
...
java_stop;

Return: end pliloadhtml;
```



› Java-Daten

pliloadhtml

```
dcl Bufferedreader_klasse type jclass;
dcl Bufferedreader_objekt type jobject init (null());
dcl Byte_array type jbyteArray;
dcl Contenttyp_string type jstring;
dcl Holen_inhalt_methode type jmethodID;
dcl Inhalt_objekt type jobject init (null());
dcl InputStream type jclass;
dcl Inputstream_objekt type jobject init (null());
dcl Inputstreamreader_klasse type jclass;
dcl Inputstreamreader_objekt type jobject init (null());
dcl Java_exception condition;
dcl Lesen type jmethodID;
dcl Messen type jmethodID;
dcl Ok type jboolean;
dcl String_objekt type jstring;
dcl Typisieren type jmethodID;
dcl Url_klasse type jclass;
dcl URL_objekt type jobject init (null());
dcl URLConnection_klasse type jclass;
dcl URLConnection_objekt type jobject init (null());
dcl Webadresse_objekt type jobject init (null());
dcl Webadresse_string type jstring;
dcl Öffnen type jmethodID;
```

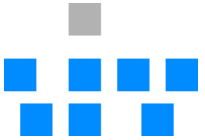


› 228 Entries

20:42

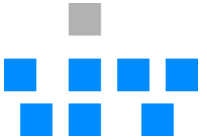
JNI

```
define structure 1 JNINativeInterface,
2 * ptr,
2 * ptr,
2 * ptr,
2 * ptr,
2 GetVersion limited entry (ptr byaddr) options (byvalue nodescriptor linkage (stdcall)) returns (byvalue type jint),
2 DefineClass limited entry (ptr byaddr, ptr byvalue, ptr byvalue, fixed bin (31)) options (byvalue nodescriptor linkage (stdcall))
returns (byvalue type jclass),
2 FindClass limited entry (ptr byaddr, nonasgn char (*) varz) options (byvalue nodescriptor linkage (stdcall)) returns (byvalue type
jclass),
2 * ptr,
2 * ptr,
2 * ptr,
2 GetSuperClass limited entry (ptr byaddr, type jclass) options (byvalue nodescriptor linkage (stdcall)) returns (byvalue type
jclass),
2 IsAssignableFrom limited entry (ptr byaddr, type jclass, type jclass) options (byvalue nodescriptor linkage (stdcall)) returns
(byvalue type jboolean),
2 * ptr,
2 Throw limited entry (ptr byaddr, type jthrowable) options (byvalue nodescriptor linkage (stdcall)) returns (byvalue type jint),
2 ThrowNew limited entry (ptr byaddr, type jclass, ptr byvalue) options (byvalue nodescriptor linkage (stdcall)) returns (byvalue
type jint),
2 ExceptionOccurred limited entry (ptr byaddr) options (byvalue nodescriptor linkage (stdcall)) returns (byvalue type jthrowable),
2 ExceptionDescribe limited entry (ptr byaddr) options (byvalue nodescriptor linkage (stdcall)) returns (byvalue ptr byvalue
optional),
2 ExceptionClear limited entry (ptr byaddr) options (byvalue nodescriptor linkage (stdcall)) returns (byvalue ptr byvalue optional),
2 FatalError limited entry (ptr byaddr, ptr byvalue) options (byvalue nodescriptor linkage (stdcall)) returns (ptr byvalue
optional),
2 * ptr,
2 * ptr, /* 20 */
2 NewGlobalRef limited entry (ptr byaddr, type jobject) options (byvalue nodescriptor linkage (stdcall)) returns (byvalue type
jobject),

. . .

2 GetJavaVM limited entry /* 219 */
(ptr byaddr, ptr byvalue) options (byvalue nodescriptor linkage (stdcall)) returns (byvalue type jint),
2 GetStringRegion limited entry /* 220 */
(ptr byaddr, type jstring, type jsize, /* Start */ type jsize, /* Länge */ ptr byvalue) options (byvalue nodescriptor linkage
(stdcall)) returns (byvalue type jvoid),
2 GetStringUTFRegion limited entry /* 221 */
(ptr byaddr, type jstring, type jsize, /* Start */ type jsize, /* Länge */ ptr byvalue)
options (byvalue nodescriptor linkage (stdcall)) returns (byvalue type jvoid),
2 * dim (6) ptr,
2 ExceptionCheck limited entry /* 228 */ (ptr byaddr) options (nodescriptor linkage (stdcall))
returns (byvalue type jboolean);
```



pliloadhtml

```
java_create_string (Webadresse_string) char (Parmadresse);
java_find_class (Url_klasse) classname ('java.net.URL');
java_create_object (Url_objekt) class (Url_klasse)
  parameter (Webadresse_string) signature('Ljava.lang.String;)V');
java_find_method (Öffnen) class (Url_klasse)
  methodname ('openStream') signature ('()Ljava.io.InputStream;');
java_call_result (Inputstream_objekt) object (Url_objekt)
  method (Öffnen) returns (jobject);
java_find_class (Inputstreamreader_klasse)
  classname ('java.io.InputStreamReader');
java_create_object (Inputstreamreader_objekt)
  class (Inputstreamreader_klasse) parameter (Inputstream_objekt)
  signature ('(Ljava.io.InputStream;)V');
java_find_class (Bufferedreader_klasse)
  classname ('java.io.BufferedReader');
java_create_object (Bufferedreader_objekt)
  class (Bufferedreader_klasse)
  parameter (Inputstreamreader_objekt)
  signature ('(Ljava.io.Reader;)V');
java_find_method (Lesen) class (Bufferedreader_klasse)
  methodname ('readLine')
  signature ('()Ljava.lang.String;');
```



› Makro ./. Aufruf

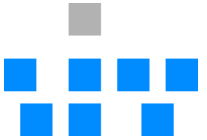
plidoma

- Makro-Aufruf:

```
java_create_object (Inputstreamreader_objekt)
   class (Inputstreamreader_klasse)
   parameter (Inputstream_objekt)
   signature ('(Ljava.io.InputStream;)V');
```

- Ergebnis:

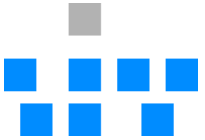
```
begin; dcl Javamethodid type jMethodID;
Javamethodid = Env.GetMethodID(JNIEnv,
   Inputstreamreader_klasse,
   '<init>', '(Ljava/io/InputStream;)V');
if Env.ExceptionCheck(JNIEnv) = JNI_TRUE
   then signal cond (Java_exception);
Inputstreamreader_objekt =  Env.NewObject(JNIEnv,
   Inputstreamreader_klasse, Javamethodid,
   Inputstream_objekt);
if Env.ExceptionCheck(JNIEnv) = JNI_TRUE
   then signal cond (Java_exception);
end;
```



› PL/I-Daten

pliloadhtml

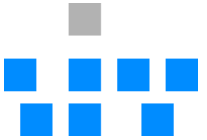
```
dcl Java_anzahl fixed bin (31);
dcl UTF_anzahl  fixed bin (31);

dcl Puffer      dim (Puffergröße) char (1) based (Puffer_ptr);

dcl Satz        char (32767) var;
dcl Präfix      fixed bin (15) based (addr(Satz));
```



› Zeilenweise lesen

pliloadhtml

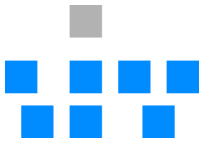
```
Pegelstand = 0;
do loop;
   java_call_result (String_objekt)
      object (Bufferedreader_objekt)
      method (Lesen) returns (jobject);
   if String_objekt = sysnull() then return;
   java_get_string_length (Java_anzahl)
      string (String_objekt);
   java_get_string_UTF_length (UTF_anzahl)
      string (String_objekt);
   if Pegelstand + UTF_anzahl > Puffergröße then do; ...  end;
   if UTF_anzahl > 32767 then do; ...  end;
   java_get_string_UTF_region (Satz) after (2)
   string (String_objekt) from (0) count (Java_anzahl);
   Präfix = UTF_anzahl;
   call HTML_korrigieren (Satz);
   call plimove (ptradd(addr(Puffer), Pegelstand),
                 addrdata(Satz), length(Satz));
   Pegelstand += length(Satz);
   end;
```



› HTML korrigieren

pliloadhtml

```
/* Übersetzen alle &-Zeichen:*/
Satz = translate(Satz, '#', '&');

/* Korrigieren fehlenden Script-Kommentar: */
K = index(Satz, '<script');
L = index(Satz, '</script');
select;
   when (In_script & L ¬= 0) do; Satz = substr(Satz, L);
      In_script = '0'b; leave; end;
   when (In_script & L = 0) do; Satz = ''; return; end;
   when (K ¬= 0 & L ¬= 0 & K < L) ;
   when (K ¬= 0 & L ¬= 0 & K > L) In_script  = '1'b;
   when (L ¬= 0)                  In_script = '0'b;
   when (K ¬= 0)                  In_script = '1'b;
   otherwise;
   end;

/* Versuch ALT-Fehler zu korrigieren: */
/* Versuch ONMOUSEOVER-Fehler zu korrigieren: */
/* Versuch INPUT-Fehler zu korrigieren: */
/* Versuch IMG-Fehler zu korrigieren: */
/* Versuch BR-Fehler zu korrigieren: */
/* Versuch <>-Fehler zu korrigieren: */
```

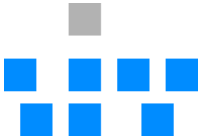


› Was haben wir bis jetzt?

20:42

pliloadhtml

- Wir möchten eine Webseite aus dem Internet lesen.
- PL/I kann das nicht direkt.
- PL/I kann die Java-Virtual-Machine aufrufen.
- Dies tut man am besten mit Makros.
- Wir lesen zeilenweise, um HTML-Fehler auszubügeln.
- Wir basteln uns ein Datenobjekt, das die Webseite enthält.
- Dieses Objekt wird durch einen Pointer repräsentiert, wenn man es mit der builtin-Funktion **`allocate()`** aufruft.
- Wir können dieses Objekt später mit **`call plifree`** freigeben.

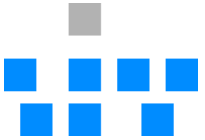


› Wie geht's weiter?

20:42

SAX ./. DOM

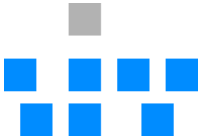
- In PL/I kann man XML parsen (nicht HTML), und zwar mit Hilfe der Builtin-Routinen **`plisaxa()`** und **`plisaxb()`**, also mit SAX (Simple API for XML).
- Eine Alternative zu SAX ist DOM (Document Object Model).
- Bei SAX werden bei jedem Antreffen eines Startkennzeichens oder eines Endekennzeichens im Datenstrom sowie bei Referenzen eigene Prozeduren aufgerufen, die dann irgendeine Verarbeitung vornehmen können.
  Vorteil: geringer Speicherbedarf,
  Nachteil: XML-Fehler fallen während der Bearbeitung auf.
- Bei DOM befindet sich das gesamte Dokument im Speicher und kann mit geeigneten Aufrufen durchstöbert werden.
  Vorteile: Das Dokument ist für eine (auch interaktive) Suche vorbereitet, die Suche kann übersichtlich programmiert werden.
  Nachteile: Kleinigkeiten an Information gehen verloren, eine zusätzliche Bearbeitungsstufe.



› Unser Hauptprogramm

Fragment 2

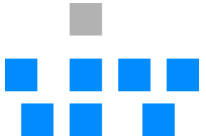
```
Programm: package options (reorder);
%include plidom;
Haupt: procedure (Parm) options (main noexecops);

dcl DOK_länge    fixed bin (31);
dcl DOK_ptr      ptr;
dcl DOMblock_ptr ptr;
...
/* HTML-Dokument in ein Objekt laden: */
call pliloadhtml(DOK_ptr, DOK_länge, Seite);
...
/* DOM-Objekt aus Web-Dokument erzeugen: */
call plidoma (DOMblock_ptr, DOK_ptr, DOK_länge);
        /* a wie bei plisaxa: aus dem Speicher */
...
call plifree (DOMblock_ptr);

end Haupt;
end Programm;
```



› Vorgefertigtes

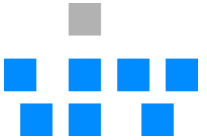
plidoma

```
define alias Event
   limited entry (pointer, pointer, fixed bin (31))
   returns (byvalue fixed bin (31))
   options (byvalue);
...
dcl 1 SAX static,
      2 ...
      2 Start_of_element type Event init
                              (SAX_Start_of_element),
      2 Attribute_name type Event init
                              (SAX_Attribute_name),
      2 Attribute_characters type Event init
                              (SAX_Attribute_characters),
...
dcl SAX_errorID   fixed bin (31);
dcl SAX_rc        fixed bin (31);
dcl SAX_string    char (32767) var;
dcl SAX_exception condition;
```



› DOM ist ein Zeiger

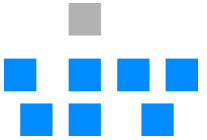

```
plidom: package options (reorder) exports (plidoma);

%include sax;

plidoma: procedure (DOMblock_ptr, DOK_ptr, DOK_länge);

%include DOMdcl;
dcl DOK_ptr   ptr;
dcl DOK_länge fixed bin (31);

Anfangsgröße = 300_000; /* Gebietsgröße */
DOMblock_ptr = allocate(16 + 16 + Anfangsgröße);
DOMblock.Größe           = Anfangsgröße;
DOMblock.Element_start   = null();
DOMblock.Element_aktuell = null();
DOMblock.Gebiet          = empty();
...
```



› DOM mit Hilfe von SAX erstellen

plidoma

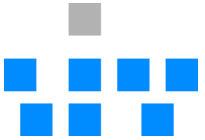
```
...
on condition (SAX_exception) begin;
   ...
   SAX_rc = 0;
   end;

SAX.Start_of_element          = DOM_starten_element;
SAX.Attribute_name            = DOM_starten_attribut;
...
SAX.Content_character_ref     = DOM_setzen_inhalt_zahl;
SAX.Attribute_predefined_ref = DOM_setzen_attribut_zeichen;
SAX.Content_predefined_ref    = DOM_setzen_inhalt_zeichen;

call plisaxa (SAX,
              DOMblock_ptr,
              DOK_ptr,
              DOK_länge);

end plidoma;
...
```



› DOMdcl.cpy (I.)

plidoma

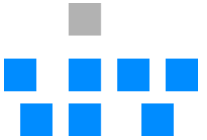
```
dcl DOMblock_ptr ptr;
dcl 1 DOMblock based (DOMblock_ptr),
      2 Element_start offset (DOMblock.Gebiet) init (null()),
      2 Element_aktuell offset (DOMblock.Gebiet) init (null()),
      2 * fixed bin (31), 2 Größe fixed bin (31),
      2 Gebiet area (Anfangsgröße refer (Größe));
dcl Element_offset offset (DOMblock.Gebiet);
dcl 1 Element based (Element_offset),
      2 Ober offset (DOMblock.Gebiet) init (null()),
      2 Unter offset (DOMblock.Gebiet) init (null()),
      2 Vorig offset (DOMblock.Gebiet) init (null()),
      2 Nächst offset (DOMblock.Gebiet) init (null()),
      2 Attribut_start offset (DOMblock.Gebiet) init (null()),
      2 Attribut_aktuell offset (DOMblock.Gebiet) init (null()),
      2 Inhalt_start offset (DOMblock.Gebiet) init (null()),
      2 Inhalt_aktuell offset (DOMblock.Gebiet) init (null()),
      2 Länge fixed bin (31),
      2 Name char (Anfangslänge refer (Länge));
...
```



› DOMdcl.cpy (II.)

plidoma

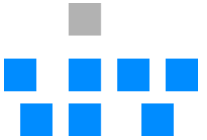
```
dcl Attribut_offset offset (DOMblock.Gebiet);
dcl 1 Attribut based (Attribut_offset),
      2 Elem offset (DOMblock.Gebiet) init (null()),
      2 Vorig offset (DOMblock.Gebiet) init (null()),
      2 Nächst offset (DOMblock.Gebiet) init (null()),
      2 Wert_start offset (DOMblock.Gebiet) init (null()),
      2 Wert_aktuell offset (DOMblock.Gebiet) init (null()),
      2 Länge fixed bin (31),
      2 Name char (Anfangslänge refer (Länge));
dcl Inhalt_offset offset (DOMblock.Gebiet);
dcl 1 Inhalt based (Inhalt_offset),
      2 Nächst offset (DOMblock.Gebiet),
      2 Länge fixed bin (31),
      2 Folge char (Anfangslänge refer (Länge));
dcl Wert_offset offset (DOMblock.Gebiet);
dcl 1 Wert based (Wert_offset),
      2 Nächst offset (DOMblock.Gebiet),
      2 Länge fixed bin (31),
      2 Folge char (Anfangslänge refer (Länge));
dcl Anfangsgröße fixed bin (31);
dcl Anfangslänge fixed bin (31);
```



› Ersatzfunktion (I.)

plidoma

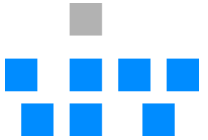
```
DOM_starten_element:
procedure (DOMblock_ptr, P, L)
   returns (byvalue fixed bin(31))
   options (byvalue);

%include DOMdcl;
dcl P ptr;
dcl L fixed bin (31);

dcl T           char (32_767) based (P);
dcl Neu_offset offset (DOMblock.Gebiet);

on area erweitern (DOMblock) mit (Element) um (100_000);
Anfangslänge = L;
allocate Element;
Element.Name = substr(T, 1, Anfangslänge);

... select; ... end; /* siehe nächste Folie */

return (0);
end DOM_starten_element;
```



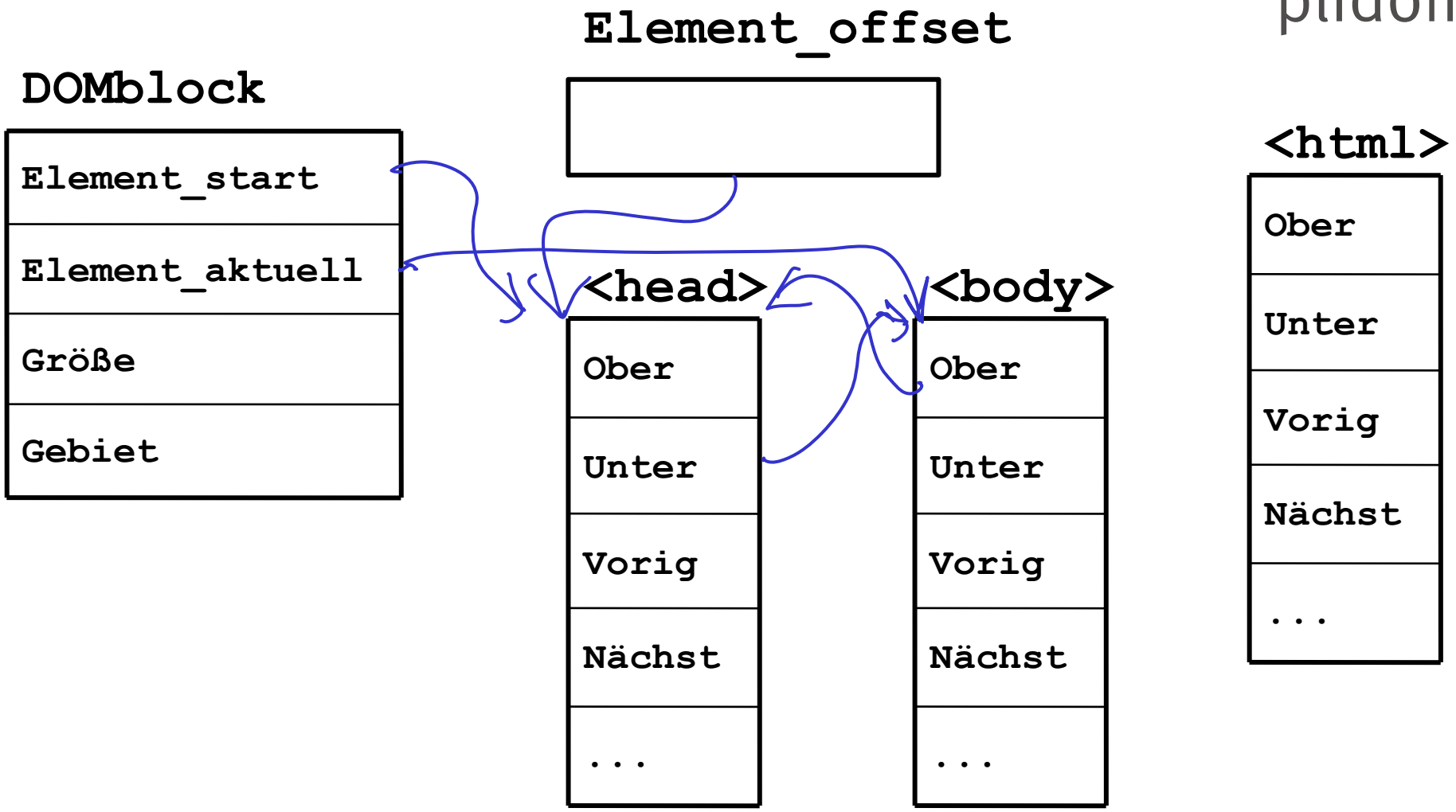
› Ersatzfunktion (II.)

plidoma

```
select;
   when (DOMblock.Element_start = null()) do;
      /* allererstes Element */
      DOMblock.Element_start = Element_offset;
      DOMblock.Element_aktuell = Element_offset;
      end;
   when (DOMblock.Element_aktuell->Unter = null()) do;
      /* erstes Unterelement */
      DOMblock.Element_aktuell->Element.Unter = Element_offset;
      Element.Ober = DOMblock.Element_aktuell;
      DOMblock.Element_aktuell = Element_offset;
      end;
   otherwise do; /* weiteres Unter-Element */
      Neu_offset = Element_offset; /* verwahren */
      do Element_offset = DOMblock.Element_aktuell->Element.Unter
      repeat Element.Nächst while (Element.Nächst ¬= null()); end;
      /* Element_offset zeigt jetzt auf das letzte Element. */
      Element.Nächst = Neu_offset;
      Neu_offset->Element.Vorig = Element_offset;
      Neu_offset->Element.ober = DOMblock.Element_aktuell;
      DOMblock.Element_aktuell = Neu_offset;
      end;
   end;
```



› Mit Papier und Bleistift ...



DOM_beenden_element:

```
DOMblock.Element_aktuell =
    DOMblock.Element_aktuell->Element.Ober;
```

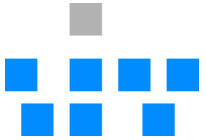
› Gebiet erweitern

- Makro-Aufruf:

```
on area erweitern (DOMblock) mit (Element) um (100_000);
```

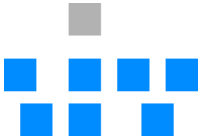
- Ergebnis:

```
on area begin;
dcl P ptr;
Anfangsgröße = DOMblock.Größe + 100_000;
allocate DOMblock set (P);
P->Element_start = Element_start;
P->Element_aktuell = Element_aktuell;
/* P->DOMblock.Größe nicht setzen !!! */
P->DOMblock.Gebiet = DOMblock.Gebiet;
free DOMblock;
DOMblock_ptr = P;
end;
```



› Was haben wir bis jetzt?

- Wir möchten aus einem im Speicher befindlichen XML-Dokument ein DOM erzeugen.
- Dazu legen wir eine PL/I-Area mit Hilfe der Builtin-Funktion **`allocate()`** an und verketten alle Blöcke im Gebiet mit Offsets miteinander.
- Ist das Gebiet voll, so legen wir ein größeres an, kopieren das alte in das neue Gebiet, geben das alte Gebiet frei und lassen den Zeiger auf das neue Gebiet zeigen.
- Das DOM kann man hinterher einfach mit **`call plifree ()`** freigeben.
- Vorteile von Gebieten:
  - Verkettete Elemente liegen im Speicher nahe beieinander, dadurch weniger "Page-Faults".
  - Freigabe durch eine Anweisung (**`free`**-Anweisung bzw. **`plifree()`**)
  - Speicherung auf Platte mit einer PL/I-write-Anweisung



› Wie geht's weiter?

pliq

- Wir haben ein DOM und können jetzt nicht mit Zeichenverarbeitung, sondern mit dem Verfolgen von Zeigern darin suchen.
- Wir brauchen am besten wieder eine Builtin-Funktion.
- In JavaScript wird ein DOM mit Funktionsaufrufen durchsucht z. B.:

```
alert(document.getElementById("abc").data);
```

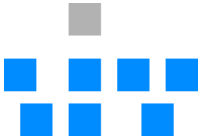
bei folgendem HTML-Code:

```
<p id="abc">Dieser Text wird ausgegeben</p>
```

- Einfacher geht es mit jQuery:

```
alert($("#abc").text());
```

- So etwas kann man nur in einer objektorientierten Programmiersprache schreiben.



› Oder???

- Wie sieht das Durchsuchen eines DOM in PHP aus?
- Mit Hilfe des Zusatzpakets QueryPath kann man etwa formulieren:

```
echo qp(DOM)->find("#abc")->text();
```

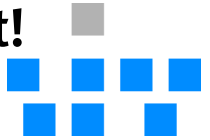
`find()` "weiß", auf welchem Objekt es arbeitet. Vor dem `->` steht es. Die Methode `find` gibt das Objekt zurück, auf dem es arbeitet!

- Auf diese Weise kann man im DOM ein Element finden, das den ID 'abc' hat, und dessen inneren Text ausgeben.
- Das sieht doch schon fast wie PL/I aus:

```
put (pliq(DOM)->find("#abc")->text());
```

- Aber PL/I ist doch gar nicht objektorientiert?!
- Behauptung:

Es reicht die Eigenschaft "objekt-basierend", Erben wird nicht benötigt!

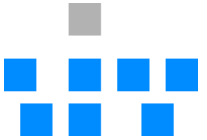


› Objekte in PL/I

- Es gibt ja die **package**-Anweisung:

```
Paket: package exports (Setzen, Holen);

dcl Zahl fixed bin (31);

Setzen_zahl: procedure (X);
dcl X fixed bin (31);
Zahl = X;
end;

Holen_zahl: procedure (X);
dcl X fixed bin (31);
X = Zahl;
end;

end;
```

- Dieses Paket ist ein Objekt, es enthält Daten und Methoden, die auf ihnen arbeiten.
- Aber was ist mit Klassen?

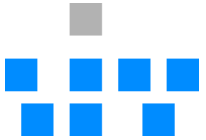


› Klassen in PL/I

pliq

- Ein Hauptprogramm, das für mehrere Objekte Methoden aufruft:

```
$test: procedure options (main);

%include Zahlobjekt;
dcl (Objekt1, Objekt2) type Zahlobjekt;

Objekt1 = Zahl_neu(14);
Objekt2 = Zahl_neu(1000);

put ($(Objekt1)->inkr(3)->wert());
call $(Objekt2)->wert(4);
put ($(Objekt1)->wert() + $(Objekt2)->wert());

call plifree (Objekt1);
call plifree (Objekt2);

end $test;
```



› Hintergründe für Klassen in PL/I

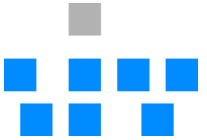
pliq

- Was wird mit **%include** eingefügt?

```
define alias Zahlobjekt ptr;
dcl $ entry (type Zahlobjekt) returns (type Zahlobjekt);
dcl Zahl_neu entry (fixed bin (31))
             returns (type Zahlobjekt);
dcl 1 * based,
      2 wert limited entry (optional fixed bin (31))
             returns (optional fixed bin (31)),
      2 inkr limited entry (fixed bin (31))
             returns (type Zahlobjekt)
      2 Zahl fixed bin (31);
```

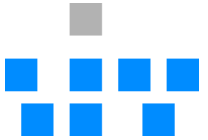
- Wesentlich:
  - Das Objekt enthält die Methoden und den Wert.
  - Die Prozedur **$** setzt den statischen Zeiger im Zahlobjekt-Paket.
  - Die Prozeduren **$**, **Zahl_neu** und **inkr** geben ein Objekt zurück.
- Auf diese Weise kann man die Funktionsaufrufe mit **->** verketten:

```
$(Objekt1)->inkr(3)->wert()
```



› Die Daten im Paket

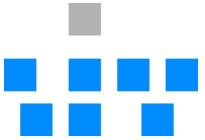

```
dcl This ptr init (null());

dcl 1 Block  static,
      2 *    limited entry (optional fixed bin (31))
             returns (optional fixed bin (31))
             init(wert_methode),
      2 *    limited entry (fixed bin (31)) returns (ptr)
             init(inkr_methode),
      2 Zahl fixed bin (31) init (0);


dcl 1 Objekt based (This),
      2 wert limited entry (optional fixed bin (31))
             returns (optional fixed bin (31)),
      2 inkr limited entry (fixed bin (31))
             returns (ptr),
      2 Zahl fixed bin (31);
```



› Die Prozeduren im Paket

pliq

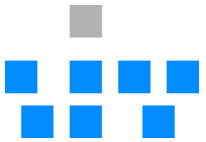
```
Zahl_neu: procedure (X) returns (ptr);

dcl X fixed bin (31);
dcl P ptr;

P = allocate(size(Block));
P->Objekt = Block;
P->Objekt.Zahl = X;
return (P);

end Zahl_neu;


$: procedure (P) returns (ptr);

dcl P ptr;

This = P;
return (This);

end $;
```



› Die Methoden im Paket

pliq

```
inkr_methode: procedure (X) returns (ptr);

dcl X fixed bin (31);

Objekt.Zahl += X;
return (This);

end inkr_methode;


wert_methode: procedure (X) returns (optional fixed bin (31));

dcl X    fixed bin (31) optional;
dcl Zahl fixed bin (31) based (This);

if present(X)
   then Objekt.Zahl = X;
   else return (Objekt.Zahl);

end wert_methode;
```



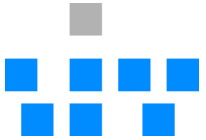
› Die Bundesliga in PHP und PL/I

20:42

pliq wie $

```
foreach (qp($seite, 'table[summary="Begegnungen"] tr:odd(1))'
                as $i -> $tr) {
   foreach ($tr->find('td') as $k -> $td) {
      if      ($k == 2) $heimverein[$i][] = $td->find('a')->text();
      elseif ($k == 4) $gastverein[$i][] = $td->find('a')->text();
      elseif ($k == 5) $ergebnis[$i][]   = $td->text(); } }
```

```
I = 1;
do Tr = pliq(pliq_ptr)->find('table[summary=Begegnungen] tr:even()')
       repeat pliq(Tr)->step() while (Tr ¬= null());
    K = 1;
    do Td = pliq(Tr)->find('td')
          repeat pliq(Td)->step() while(Td ¬= null());
       select (K);
          when (3) Heimverein(I) = pliq(Td)->find('a')->text();
          when (5) Gastverein(I) = pliq(Td)->find('a')->text();
          when (6) Ergebnis(I)   = pliq(Td)->text();
          other; end;
       K += 1; end;
    I += 1; end;
```

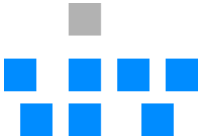


› Das pliq-Objekt

- Analog zum Zahlobjekt:

```
dcl 1 Pliq_objekt   based (This),
      2 step         limited entry () returns (ptr),
      2 find         limited entry (char (*) var) returns (ptr),
      2 text         limited entry (char (*) var optional)
                     returns (char (32767) var),
      2 attr         limited entry (char (*))
                     returns (char (32767) var),
      2 DOMblock_ptr     ptr,
      2 Durch_find_angelegt bit,
      2 Tab_aktuell fixed bin (31),
      2 Anzahl      fixed bin (31),
      2 Tab         dim (Anfangsanzahl refer (Anzahl))
                    offset (DOMblock.Gebiet);
```

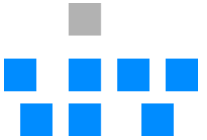
- Wenn per **`find()`** mehrere Elemente gefunden werden, werden diese in **`Tab`** abgelegt.



› Unser Hauptprogramm

Fragment 3

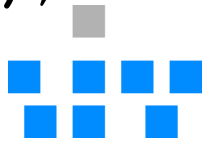
```
Programm: package options (reorder);
...
Haupt: procedure (Parm) options (main noexecops);
...
/* HTML-Dokument in ein Objekt laden: */
call pliloadhtml(DOK_ptr, DOK_länge, Seite);
...
/* DOM-Objekt aus Web-Dokument erzeugen: */
call plidoma (DOMblock_ptr, DOK_ptr, DOK_länge);
...
/* pliq-Objekt erzeugen und mit DOM-Objekt verbinden: */
pliq_ptr = pliqalloc(DOMblock_ptr);
...
/* pliq-Objekt mit pliq durchstöbern: */
... /* vorige Folie */
/* Ausgabe: */
... /* nächste Folie */
end Haupt;
end Programm;
```



› Die Ausgabe

pliq

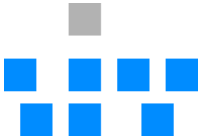
```
dcl Gast        char (100);
dcl Gastverein dim (9) char (100) var;

do I = 1 to 9;
   parse value (Ergebnis(I))
         with (Endergebnis, '#nbsp;', Halbzeitergebnis);
   Heim = '';
   Lh = memconvert(addr(Heim), length(Heim), 850, /* PC */
                   addrdata(Heimverein(I)),
                   length(Heimverein(I)), 1208); /* UTF-8 */
   Gast = '';
   Lg = memconvert(addr(Gast), length(Gast), 850,
                     addrdata(Gastverein(I)),
                     length(Gastverein(I)), 1208);
   Beides = Endergebnis || ' ' || Halbzeitergebnis;
   put edit (left(Heim, Lh), '-', left(Gast, Lg), Beides)
     (col (1), a, col(22), a, col(25), a, col(46), a);
   end;
```



› Abklappern mit Makros

pliq

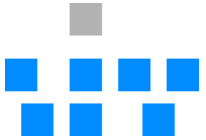
```
do_foreach (Element);
   if Element.Name = Aufruf.Parm then select; /* Heureka */
      when (letzter(Aufruf)) do;
         anlegen (Fund);
         Fund.Stelle = Element_offset;
         end;
      when (Aufruf.Nächst->Aufruf.Hier)
         call Aufruf.Nächst->Aufruf.Prog
            (DOMblock_ptr, (Zähler), (Element_offset),
            (Aufruf.Nächst), Aufrufblock_ptr, Fundblock_ptr);
      otherwise
         if Element.Unter ¬= null() then
            call Aufruf.Nächst->Aufruf.Prog
               (DOMblock_ptr, (Zähler), (Element.Unter),
               (Aufruf.Nächst), Aufrufblock_ptr, Fundblock_ptr);
      end;
   else if Element.Unter ¬= null() then
      call Aufruf.Prog
        (DOMblock_ptr, (Zähler), (Element.Unter),
        (Aufruf_offset), Aufrufblock_ptr, Fundblock_ptr);
   end;
```



› do_foreach-Makro

pliq

```
%do_foreach:
procedure (X, From) statement;

dcl X char;
dcl From char;

if parmset(From)
   then ans ("do " || X || "_offset = " || From)
            skip noscan;
   else ans ("do " || X || "_offset = " || X || "_start")
            skip noscan;
ans ("   repeat " || X || ".Nächst") skip noscan;

   ans ("   while (" || X || "_offset ¬= null());")
   skip noscan;

%end do_foreach;

%dcl do_foreach entry;
```

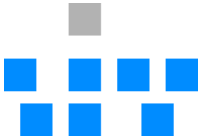


› Konjugieren in PL/I

20:42

pliq

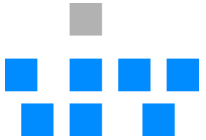
```
%letzter:
letzte:
letztes:
procedure (X) returns (char);

dcl X char;

return ('(' || X || ".Nächst = null())");

%end letztes;

%dcl letzter entry;
%dcl letzte entry;
%dcl letztes entry;
```



› anlegen-Makro

pliq

```
%anlegen:
procedure (X) statement;

dcl X char;

ans ("do;")                                              skip noscan;
ans ("allocate " || X || ";")                            skip noscan;
ans ("if " || X || "_start = null()")                    skip noscan;
ans ("then " || X || "_start = " || X || "_offset;")     skip noscan;
ans ("if " || X || "_aktuell = null()")                  skip noscan;
ans ("then " || X || "_aktuell = " || X || "_offset;")
                                                         skip noscan;
ans ("else " || X || "_aktuell->" || X || ".Nächst"
    || " = " || X || "_offset;")                         skip noscan;
ans (X || "_aktuell = " || X || "_offset;")              skip noscan;
ans ("end;")                                             skip noscan;

%end anlegen;

%dcl anlegen entry;
```



› Zusammenfassung

pliloadhtml – plidoma – pliq

- Drei BUILTIN-Funktionen gebastelt
- Mit Java ins Internet
- XHTML ist selten korrektes XML
- Mit **`plisaxb()`** ein DOM erstellt
- Objekte als Zeiger mit **`allocate()`** und **`plifree()`**
- Am Zeiger hängt eine AREA-Variable (Lokalität)
- Klassen als Pakete mit statischem Zeiger
- Methoden als Paketprozeduren
- Funktionen verketten mit **`returns (ptr)`**
- Komplexe Aufrufe mit Makros