Matthias Jauernig
03IN1
07.07.05
www.linux-related.de

YADS – *Y*et *A*nother *D*IAS *S*ummary

Kleine unvollständige Zusammenfassung der wichtigsten Funktionen und Operationen beim Arbeiten mit DIAS bzw. der Programmiersprache LAMBDA zur Prüfungsvorbereitung im Fach „Digitale Bildverarbeitung“, HTWK Leipzig.

1. **elementare Syntax:**
   - Allgemein gilt: Variablen müssen nicht deklariert werden, dies geschieht implizit durch Zuweisungen an sie.
   - Schlüsselwörter sind nicht „case sensitive“, es ist also egal, ob man z.B. while oder WHILE schreibt.
   - Bei einstelligen Operatoren gilt die prefix-Schreibweise, also muss man z.B. für den Absolutbetrag von -1 schreiben: „-1 abs“.
   - Mathematische Klammern werden mit {} angegeben, () bilden die Klammern zum Abgreifen eines Pixels aus einem Bild.
   - *Umgang mit den Akkumulatoren:*
     - +term=var → gesamte Rechnung mit Integerzahlen durchführen, Ergebnis als Intergerzahl abspeichern
     - "+term=var → führe gesamte Rechnung mit Gleitkommazahlen durch, speichere Ergebnis als Gleitkommazahl
     - +term1"=var → rechne term1 mit Integer-Zahlen, speichere das Ergebnis aber als Gleitkommazahl ab
     - +term1"/term2'*term3"=var → rechne term1 mit Integer-Zahlen, schiebe das Ergebnis dann in den Gleitkomma-Akku und dividiere dieses mit term2, multipliziere dann das Zwischenergebnis im Integer-Akku mit term3 und speichere das Endergebnis als Gleitkommazahl in var ab
     - Bsp.: "+11/2=var; fout var → 5.5
     - Bsp.: +11"/2=var; fout var → 5.5
     - Bsp.: +11/2"=var; fout var → 5.0 (Rechnung im Integer-Akku durchgeführt)
     - ACHTUNG: "+11/2=var; fout var; +2=var; fout var;
       → Dieses Schnipsel würde zur Ausgabe von 5.5 5.5 führen und nicht wie erwartet zu 5.5 2; Grund: das Gleitpunkt-var ist eine andere Variable als das Integer-var und da beide Male eine Gleitpunktzahl ausgegeben wird, wird hier nur auf das Gleitpunkt-var zugegriffen.
   - *if-Anweisung:*
     IF bed THEN anw1 ELSE anw2 ENDIF
     → wenn bed erfüllt, dann führe anw1 aus, anw2 sonst
     IFN bed THEN anw2 ELSE anw1 ENDIF
     → wenn bed nicht erfüllt ist, dann führe anw2 aus, anw1 sonst

- *wloop-Schleife:*
  WLOOP name x y x1 y1 x2 y2
  anw
  END name
  → Es wird der Anweisungsblock anw für jeden Bildpunkt im aktuellen Fenster durchlaufen, wobei die Schleife den Namen name besitzt. Optional kann ein rechteckiger Fensterbereich von (x1,y1) bis (x2,y2) angegeben werden.
- *loop-Schleife:*
  LOOP name var startw endw schritt
  anw
  END name
  → Es wird die Anweisungsfolge anw endw-startw+1 mal ausgeführt, wobei var alle Werte zwischen startw und endw durchläuft. Will man um einen anderen Wert als 1 erhöhen, so kann man dies optional über schritt angeben. name ist der Schleifenname. LOOP kommt somit einer benannten for-Schleife gleich.
- *while-Schleife:*
  WHILE bed
  anw
  ENDWHILE
  → Solange die Bedingung bed erfüllt ist, wird der Anweisungsblock anw ausgeführt.
- *for-Schleife:*
  FOR var startw endw schritt
  anw
  ENDFOR
  → Die Variable var wird vom Startwert startw bis zum Endwert endw inkrementiert und der im Schleifenkörper enthaltene Anweisungsblock anw wird jedes Mal ausgeführt. Optional kann anstelle einer Schrittweite von 1 auch eine andere Inkrementierung +schritt angegeben werden.

2. **Fenster-Operationen:**
   - (x,y,nr)
     → Abgreifen des Grauwertes des Pixels (x,y) im Bild nr (Bsp.: +(x,y,1)=anz) bzw. Setzen dessen bei einer Zuweisung. (Bsp.: +wert=(x,y,1)).
   - PSIZE x y anz
     → Stellt anz Bilder mit der max. Ausdehnung von je x*y Pixeln bereit. Wenn anz und y weggelassen werden, werden max. viele Bilder mit x*x Pixeln erzeugt.
   - MON mode
     → Realisierung von Visualisierungen: mode=0: Darstellung der Grafik auf dem Grafikmonitor aus (Standard), mode=1: Darstellung der Grafik auf dem Grafikmonitor ein, mode=8: Aktualisierung des Grafikbildschirms.
   - INFPSIZE x y
     → Schreibt die gerade benutzte Fenstergröße in die Variablen x und y.
   - TEXT t x y groesse wert nr
     → Gibt den Text t ab der Position (x,y) (linke obere Ecke) mit der Größe groesse und dem Grauwert wert im Bild nr aus. Weitere Parameter sind im DIAS-Handbuch nachzulesen.

3. **Ein-/Ausgaben:**
   - Eingaben:
     - INP var1 var2 …
       → Es wird mind. eine Integer-Variable in der Shell vom Benutzer eingelesen, ab var2 optional.
     - FINP var1 var2 …
       → In der Shell wird eine Gleitkomma-Variable vom Benutzer eingelesen, ab var2 optional.
     - PINP name nr
       → Die Bilddatei mit dem Namen name wird in das Bild nr eingelesen. Werden name und nr nicht angegeben, so erfolgt eine Bildauswahl vom Benutzer. Gibt man name=^^ an, so erfolgt die Bildwahl vom Benutzer, das gewählte Bildfile wird dann im Bild nr angezeigt.
     - PINPRGB name r g b
       → Die Bilddatei name wird geöffnet und die Farbauszüge in den Bildern r g b dargestellt. Gibt man name=^^, so erfolgt eine Bildwahl vom Benutzer.
   - Ausgaben:
     - OUT var1 var2 …
       → In der Shell wird mind. eine Integer-Variable ausgegeben, ab var2 ist die Angabe optional.
     - FOUT var1 var2 …
       → In der Shell wird mind. eine Gleitkomma-Variable ausgegeben, ab var2 ist die Angabe optional.
     - POUT name nr
       → Das Grauwert-Bild im Fenster nr wird in der Datei name gespeichert. Wird name=^^ angegeben, so wählt der Benutzer ein Bild. Bei Weglassen von name und nr wählt der Benutzer eine Datei und das Grauwertbild im aktuellen Fenster wird ausgegeben.
     - POUTRGB name r g b
       → Das Farbbild mit den RGB-Auszügen in den Fenstern r, g und b wird in der Datei name gespeichert. Gibt man name=^^ an, so muss die Bilddatei vom Benutzer gewählt werden.

4. **Bildverarbeitung:**
   - PCLR nr
     → Setzt die Grauwerte im Bild nr auf 0.
   - PSET grw nr x1 y1 x2 y2
     → Setzt die Bildpunkte im aktuellen Bild bzw. dem mit der Nummer nr auf den Grauwert grw unter optionaler Angabe eines rechteckigen Bereichs.
   - PMOV nr1 nr2 x1 y1 x2 y2
     → Kopiert das Bild nr1 in das Bild nr2 bzw. optional nur einen rechteckigen Bereich davon.
   - PINV nr1 nr2 x1 y1 x2 y2
     → Invertiert das Bild nr1 und stellt das Ergebnis in Bild nr2 dar oder in nr1, falls nr2 nicht angegeben wurde. Bei Angabe von x1, y1, x2 und y2 wird nur dieser rechteckige Bereich invertiert.
   - PAND nr1 nr2 nr3 x1 y1 x2 y2
     → Führt ein bitweises UND aller Pixel bzw. der Pixel im optional angegebenen rechteckigen Bereich von Bild nr1 und Bild nr2 durch und speichert das Ergebnis in Bild nr3.

- POR nr1 nr2 nr3 x1 y1 x2 y2
  → Wie PAND, nur mit bitweisem ODER.
- PXOR nr1 nr2 nr3 x1 y1 x2 y2
  → Wie PAND, nur mit bitweisem XOR.
- RGBHSI r g b h s i
  → Konvertiert das RGB-Bild mit den Grauauszügen in den Bildern r, g und b in HSI und stellt das Ergebnis entweder auf sich selbst oder in den Bildern h, s und i dar.
- HSIRGB h s i r g b
  → Konvertiert das HSI-Bild mit den Grauauszügen in den Bildern h, s und i in RGB und stellt das Ergebnis entweder auf sich selbst oder in den Bildern r, g und b dar.
- SHOWRGB r g b
  → Stellt das RGB-Bild mit den Grauauszügen in den Bildern r, g und b als Farbbild auf dem Bildschirm dar. Wird SHOWRGB ohne Parameter aufgerufen, so wird wieder die Grauwert-Fensterdarstellung angezeigt.

5. **Zeichnen geometrischer Figuren:**
   - LINE x1 y1 x2 y2 grw nr
     → Zeichnen einer Linie von (x1,y1) nach (x2,y2) mit dem Grauwert grw im Bild nr.
   - CIRCLE xm ym r grw nr mode
     → Zeichnen eines Kreises mit dem Mittelpunkt (xm,ym) und dem Radius r mit dem Grauwert grw im Bild nr. Wird mode angegeben, so wird der Kreis bei mode=2 ausgefüllt, bei mode=1 nur die Umrandung gezeichnet (Standard).
   - FRAME xm ym a b wert nr mode
     → Zeichnen eines Rechtecks um den Mittelpunkt (xm,ym) mit den Seitenlängen a und b, dem Grauwert wert im Bild nr. Wenn mode angegeben wird, so wird das Rechteck bei mode=2 ausgefüllt, bei mode=1 wird nur die Umrandung gezeichnet (Standard).

6. **Integer-Operatoren:**
   - Einstellige Integer-Operatoren:

Operator	Operation	Resultat
abs	Absoluter Betrag	Integer
chr	Zeichen-Umwandlung	String
neg	Negation	Integer
sign	Vorzeichen	Integer
sqr	Quadrat	Integer
sqrt	Quadratwurzel	Integer
"	Umwandl. in Gleitkomma	Gleitkomma
^	Umwandlung in String	String
#	Register-Zugriff	Integer

- Zweistellige Integer-Operationen (Resultat immer Integer):

+	Addition	min	Minimum
-	Subtraktion	mod	Modulo
*	Multiplikation	or	Bitweises ODER
/	Division	pyth	sqrt(IAC^2+Operand^2)
and	Bitweises UND	xor	Bitweises Exklusiv-ODER
max	Maximum	pwr	Potenz
#()	Byte im Reg.feld		

7. **Gleitkomma-Operatoren:**
   - Einstellige Gleitkomma-Operatoren:

abs	abs(FAC)
atan	arctan(FAC)
acos	arccos(FAC)
asin	arcsin(FAC)
cos	cos(FAC)
exp	e^(FAC)
fract	fract(FAC)
int	int(FAC)
inv	1/(FAC)
ln	ln(FAC)
neg	-(FAC)
sign	sign(FAC)
sqr	(FAC)*(FAC)
sqrt	sqrt(FAC)
tan	tan(FAC)
'	int(FAC)
^	string(FAC)

   - Neben +, -, *, / gibt es weitere zweistellige Gleitkomma-Operatoren:

log	log(FAC,Operand) -> FAC
max	max(FAC,Operand) -> FAC
min	min(FAC,Operand) -> FAC
pyth	sqrt(FAC^2+Operand^2) -> FAC
pwr	(FAC)^Operand -> FAC

8. **Register:**
   - TEMPREG groesse name
     → Legt ein neues Register mit groesse Elementen an, das erste Element kann man dabei über name abgreifen.
   - RCLR pos anz
     → Ab Position pos werden anz Elemente im Register gelöscht (0 gesetzt).
   - #{name+i}
     → Zugriff auf die i. Position eines Register, wenn name die erste Position angibt. Schreibender Zugriff z.B. über +wert=#{name+i}, lesender Zugriff z.B.: +#{name+i}=wert.

9. **Markierungswerte:**
   - [x,y,nr]
     → Ansprechen von (ganzzahligen) Markierungswerten, um den markierten Wert zu erhalten (z.B. +[x,y,nr]=wert) bzw. zu setzen (z.B. +wert=[x,y,nr]).
     Folgende Farben sind standardmäßig den Markierungswerten zugeordnet:

0	[unmarkiert]	4	gelb
1	rot	5	kobaltblau
2	grün	6	violett
3	blau	7	rosa

10. **Zufallszahlen:**
   - RNDM m Z1 … Z9
     Erzeugt ganzzahlige Zufallszahlen im Bereich von 0 bis m und speichert diese in den Integer-Variablen Z1 bis Z9 ab (optional ab Z2).
   - FRNDM m sigma Z1 … Z9
     Erzeugt Gleitkomma-Zufallszahlen mit dem Mittelwert m und einer Standardabweichung sigma und stellt diese in den Gleitkomma-Variablen Z1 bis Z9 bereit (optional ab Z2).