

Definitionsmodul: Datei : name.od
DEFINITION Hallo;
END Hallo.

Modul: Datei: name.om
MODULE Hallo;
BEGIN
END Hallo.

Deklaration:
IMPORT Read, Write;
CONST x = y;
eps = 1.0E-6
TYPE (unten)
VAR x : INTEGER;
string : ARRAY 256 OF CHAR;

IMPORT Read
Write
Streams (z.B. stdin, stdout, stderr)
ASCII
Math
SysStat
Args := UnixArguments;
UF:=UnixFiles;
Process
Conclusions (Bewirkt ausführlichere Fehlermeldungen)

Datentypen BOOLEAN (FALSE oder TRUE)
CHAR
SHORTINT
INTEGER
LONGINT
REAL (1.234)
LONGREAL
STRINGS
SET (Ganze Zahlen von 0 bis max ; Mengen z.B. x = {1,5,8})
ARRAY size OF datentyp; (Zugriff auf mehrdim. ARRAYs: array[1,1])
Streams.Stream (Streams importieren!)
BYTE
POINTER TO (Typ aus ARRAY oder RECORD)
RECORD bestehend aus Elementen versch. Typs

TYPE Vektor = ARRAY length OF INTEGER; ([0, .. ,(length-1)] **Achtung !!!**
Matrix = ARRAY length , length OF INTEGER; **Anders bei Prozeduren**
Rec = RECORD
element:INTEGER;
END;
Funktion = PROCEDURE(x:REAL):REAL; (Skript S.139)
POINTER TO typ (typ vom Typ Array oder Record)

IF: IF 1#1 THEN
....
ELSIF ... END;
ELSE
END;

CASE CASE x OF (hier: x : CHAR;)
| 'a' .. 'z' : Write.Ln;
| '#', '+', '5' : Write.String("Hello World");
ELSE
END;

REPEAT	REPEAT UNTIL 1 # 1 ;
---------------	----------------------------------

LOOP	LOOP EXIT; END;
-------------	--------------------------------

WHILE	WHILE (x = y) & ~ (FALSE) DO INC(x); DEC(x); END;
--------------	---

PROCEDURE	hallo(x , y : INTEGER ; ch : CHAR ; VAR s : ARRAY OF CHAR) : INTEGER; CONST TYPE VAR PROCEDURE BEGIN RETURN; RETURN n; END hallo; Achtung: in 1. Zeile bei Vektoren oder Matrizen keine Länge vector : ARRAY OF CHAR matrix: ARRAY OF ARRAY OF INTEGER Call by referenz (VAR) - Original wird bearbeitet Call by value - Kopie wird bearbeitet Längenbestimmung der Vektoren/Matrizen: length := SHORT(LEN(vector)); (LEN liefert LONGINT und SHORT conv. in INTEGER) dim1 := SHORT(LEN(matrix,0)); dim2 := SHORT(LEN(matrix,1)); Rückgabewert keine Arrays oder Records. Prozeduren, die eine Funktion bekommen: PROCEDURE(x:Funktion;y,z:INTEGER); (Funktion siehe oben) (Skript S.139) Aufruf: :(sin,y,z); → Bei diesem Typ kein Wert an Prozedur weitergereicht
------------------	---

Write:	Write.Ln; (schreibt newline) Write.Line("dies ist ein String"); (ganze Zeile) Write.String("String"); Write.Int(x,width); Write.Real(x,width); Write.StringS(Streams.stderr, "falsche Eingabe"); Streams Write.LnS(Streams.stderr); (Ausgabe der Fehlermeldung) Streams
---------------	---

Read:	Read.Line(line); (zum einlesen incl.Newline) Read.Int(n); Read.Char(ch); Read.Real(x); Read.IntS(s:Streams.Stream;VAR int:INTEGER); (Definition) Streams Read.CharS(Streams.Stdin , ch); (Liest aus Stream in Variable) Streams
--------------	--

Streams

Benutzen der „S“ – Funktionen von Read und Write

~Streams.stdin.eof

(Nicht Ende von Input)

Streams.ReadByte(s:Streams.Stream;VAR byte:BYTE):BOOLEAN;

(Byteweise lesen)

Streams.WriteByte(s:Streams.Stream;byte:BYTE):BOOLEAN;

(Byteweise schreiben)

Streams.Seek(s:Streams.Stream;offset:Count;whence:Whence):BOOLEAN;

(Positionieren)

(Count vom Typ LONGINT | Whence vom Typ SHORTINT)

Streams.Tell(s:Streams.Stream;VAR offset:Count):BOOLEAN;

(Position bestimmen)

Streams.Close(s:Streams.Stream):BOOLEAN;

(Schließen einer Verbindung)

Streams.stdin.count

Streams.Stream.count

(Hat Wert 1, falls Zeichen (# NewLine...) gelesen, ansonsten 0, LongInt)

UnixFiles

IMPORT UF:=UnixFiles;

UF.Open(VAR s:Streams.Stream;filename:ARRAY OF CHAR;

mode:Mode;bufmode:Streams.BufMode;errors:RelatedEvents.Object):BOOLEAN;

Mode vom Typ SHORTINT

→ read=0 , write=1 , rdwr=2 , create=4, condcreate=8 (nur wenn nicht schon da)

→ UF.Open(stream,filename,mode,1,NIL);

Beispiele: Erzeugen und schreiben in Datei (d.h. in outstream)

IF ~UF.Open(outstream,outfile,1+4,1,NIL) THEN

Write.StringS(Streams.stderr,"can't open");

Write.StringS(Streams.stderr,outfile);

Write.LnS(Streams.stderr);

Process.Exit(2);

END;

Kopieren

WHILE Streams.ReadByte(instream,b) DO

(b vom Typ BYTE)

IF ~Streams.WriteByte(outstream,b) THEN

Process.Exit(n);

END;

END;

Schließen

IF ~Streams.Close(instream) OR ~Streams.Close(outstream) THEN

Write.StringS(Streams.stderr,"Error at close");

Write.LnS(Streams.stderr);

Process.Exit(n);

END;

Zeiger:

TYPE VectorPtr = POINTER TO Anothertype;

(Anothertyp vom Typ ARRAY oder RECORD)

VAR p:VectorPtr;

NEW(p);

(Speicherplatz reservieren und Zeiger auf neues Objekt erstellt)

p:=NIL;

(Nirvana)

Seien p1 und p2 Zeiger

=> p1:=p2

(Es werden "Zeiger zugewiesen", d.h. beide zeigen auf identisches Objekt)

=> p1^:=p2^

(Es wird das "verknüpfte Objekt" zugewiesen, d.h. p1 arbeitet auf Kopie vom p2- Objekt)

Zugriff: p.element

(greift auf Inhalt des Records an der Stelle Element zu, oder wenn Element ist wieder ein Pointer dann auf Objekt, auf den er zeigt)

Bei Fehlern bitte eine Mail an :

sascha@wirtschaftsphysik.de

andreas@wirtschaftsphysik.de

Operatoren :	IN	(Element der Menge ?)
	BOOLEAN	
	IS	(Typ-Test)
	ASSERT (i # 0);	(Darf nicht 0 werden/sein)

<u>Unix - Shell – Befehle</u>		
Commando	< datei	(stdin aus Datei)
	> datei	(stdout auf Datei (wird gelöscht))
	>> datei	(stdout an Datei anhängen)
	2> datei	(stderr auf Datei (wird gelöscht))
	2>> datei	(stderr an Datei anhängen)

<u>Typ-Erweiterung</u>		
Typus1 = RECORD		
	x,y : INTEGER;	
END;		
Typus2 =RECORD (Typus1)		(Typus1 um z erweitert)
	z : INTEGER;	
END;		
VAR	basis : Typus1;	
	erw : Typus2;	
Zugriff:	erw.x := 1;	
	erw.z := 2	
	erw.z := 2;	
	basis := erw;	(Erlaubt)
	erw := basis;	(LAUFZEITFEHLER !!!!)
(Es müssen alle Elemente des Typs richtig belegt werden)		
PROCEDURE test(a:Basistyp);		
	Der statische Typ von a ist "Basistyp"	
	Der dynamische Typ von a ist der zur Laufzeit bestimmte Typ.	
	Der dynamische Typ muß eine Erweiterung des statischen Typs sein.	
	Zum Testen zur Laufzeit:	
IF a IS Erweiterungstyp THEN ...		

<u>Objects</u>	IMPORT Objects;
	General=Objects.Object;
	ElementRec=RECORD(Objects.ObjectRec)
	count:INTEGER;
	content:General; (Somit kann ein beliebiger Typ übergeben werden)
	END;

<u>Heterogene Strukturen S.254ff</u>	
Strings.Open(stream,line);	(Liest line in Stream)
Streams.stdin.count	(1, wenn beim letzten lesen (egal ob Stream oder nicht) ein Zeichen ungl. Newline gelesen wurde)
DEFINITION ...	
	CONST
	TYPE
	PROCEDURE
END dhhf.	