- ADA-Schnellübersicht -

Funktion

```
function Name(Variable : in|out Typ; ...) return Typ is
   - - Typen, Variablen, Funktionen, Prozeduren
begin
    Anweisung; ...
   return Wert;
end Name;
```

```
function test(a:in integer) return boolean is
   a : character := 'X'; b : integer := 2;
begin
   a:=3;
   return true;
end test;
```

Prozedur

```
procedure Name(Variable : in|out Typ; ...) is
   - - Typen, Variablen, Funktionen, Prozeduren
begin
   Anweisung; ...
end Name;
```

```
procedure test(a:in integer; b: in out boolean) is
   a : constant integer := 2;
begin
   ...;
end test;
```

Task

```
task TaskName is
   entry EntryName1;
   entry EntryName2(Variable : in|out Typ; ...);
   ...
end TaskName;

task body TaskName is
   - - Typen, Variablen, Funktionen, Prozeduren
begin
    ...
   accept EntryName1;
   - - bleibt am accept hängen bis entry-Aufruf
    ...
   accept EntryName2(Variable : in|out Typ; ...) do
      - - Kritischer Abschnitt, kein Taskwechsel !!!
   end EntryName2;
    ...
end TaskName;
```

```
task t is
   entry e1;
   entry e2(x : in integer);
   ...
end t;

task body t is
      a : integer;
begin
   a := 0;
   accept e1;
   Put(''Task gestartet!'')
   ...
   accept e2(x : in integer) do
     tue_wichtige_sache;
   end e2;
    ...
end t;
```

Grundstruktur (Beispiel)

```
with PACKAGE_NAME;  use PACKAGE_NAME;

procedure Haupt is
   -- Typdeklarationen
   type neu_int is new integer range 10..20;
   subtype sub_int is integer range 0..9;

   -- Variablendeklarationen
   ni : neu_int := 11;

   -- Unterfunktionen/-Prozeduren
   procedure p1(a:in integer; b:in boolean) is
   begin
      -- Anweisungen Prozedur
   end p1;

begin
   -- Anweisungen Hauptprogramm
end Haupt;
```

Kontrollstrukturen

if-then	while-do	repeat-until	for-to-do

`if` Bedingung `then`
Anweisung; ...
`else`
Anweisung; ...
`end if;`
	`while` Bedingung `loop`
	Anweisung;
	...
	`end loop;`
		`loop`
		Anweisung;
		...
		`exit when` Ausdruck;
		`end loop;`
			`for` Variable `in` Range `loop`
			Anweisung;
			...
			`end loop;`

FOR-Schleife: Variable wird automatisch für die Dauer des Schleifendurchlaufs deklariert (hat den Typ, den Range vorgibt).
Range ist entweder `(`min`..`max`)` oder ein Arrayname.

Variablendeklaration

Name, ... : Typ;	`a,b,c : integer;`
Name :Typ := Startwert;	`A : integer :=5;`
Name `:constant` Typ;	`A : constant integer;`

Arrays

Name `: array` Range `of` Typ;	`a : array(1..9) of float;`
Name `:=(others=>`Wert`);`	`a := (others => 0.0);`
Name`(`Index`);`	`a(i);`

Variablen existieren immer nur innerhalb des Blockes (Funktion, Schleife ...) in dem sie deklariert wurden.
Zusätzliche Attribute von Arrays sind `a'first`, `a'last` und `a'range`. Range ist entweder `(`min`..`max`)` oder ein Aufzählungstyp.

Records		Typdeklaration		Variablen (Basistypen)	

Name : record
  Varname : Typ;
  ...
end record;
	r : record
	  i : integer;
	  b : BIT_Typ;
	  f : float;
	end record;
		type Name is Vartyp;	type t is record
			  i : integer;
			  f : float;
			end record;
				boolean	true

				character	'A'
				integer	12
		subtype Name is Typ range (min..max);	subtype s is integer range (0..9)	float	0.3

Subtypen werfen eine **CONSTRAINT_ERROR** Exception wenn ihr Bereich überschritten wird.