Schleifen
- "loop"
- Abbruch exit, next

bedingungslose Schleife
```
<label> : loop
        .
        .
        .
        end loop [<label>];
```

```
elementar_loop : loop
                 if i = 31 then
                          exit;
                 else
                 i := i+1;
                 q(i) <= '0';
                 end if;
                 end loop elementar_loop;
```

for-Schleifen
```
<label> : for <index> in <range> loop
                .
                .
                .
                end loop [<label>];
```
Bsp.:
```
for i in 0 to 31 loop
        addr(i) <= '0';
end loop;
```

```
<label> : while <condition> loop
        .
        .
        .
        end loop;
```
Bsp.:
```
while i<5 loop
        q(i) <= data_in(i)
        i := i + 1;
end loop;
```

optionales & zusätzliches
```
[<label>] exit [<label>] [when]:
          next
```

Unterprogramme (Subprograms)
- function, procedure
- Operatoren, Datentypen, Konstanten, Konvertierung
- lokal (in einer architecture), global (package)
- rein sequenzielles VHDL
- laut IEEE_1076.6 sind Signaldeklarationen nicht synthesefähig
- Rekursion - nicht synthesefähig!

Funktionen
- mehrere Übergabeparameter
- ein Rückgabewert

```
function <func_id> (<parameter_list>) return <type_id> is
        <variable. constant, type declaration>;
begin
        <ssequential statements>
return <id>;
end [<func_id>];
```

Bsp.:
```
funtion vec2int (v:std_logic_vector) return integer is
        variable tmp: integer := 0;
begin
        for i in v'range loop
                tmp : = tmp*2;
                if v(i) = '1' then
```

```
                        tmp := tmp + 1;
                end if;
        end loop;
        return tmp;
end vec2int;

(int_var := vec2int(vec_var))

function overloading

type tri_logic is ('0', '1', 'K');
function and (in_1: tri_logic; in_2: tri_logic) return tri_logic is
begin
        if in_1 = '1' and  in_2 = '1'then return '1';
        elsif in_1 = '0' or in_2 = '0' then return '0';
        else return 'K';
        end if;
end and;

Zustände von Funktionen: pure, impure           -- nicht für HBS relevant

Prozeduren
- liefern keine Rückgabewerte: => nicht in Ausdrücken
- mehrere Ü-Parameter möglich (in, out, inout)
[in = ro]
[out = wo]
[inout = r/w]

procedure <proc_id> (<param_list>] is <declaration>
begin
        <sequ. VHDL>
end [<proc_id>];

Bsp.:
procedure std_logicvec2int (v: in std_logic_vector;
                            f: out boolean;
                            result: out int) is
variable tmp : integer := 0;
begin
        result := 0;
        f := true;
        for i in v'range loop
                result := result*2;
                if v(i) = '1' then
                        result := result + 1;
                        f := false;
                end if;
        end loop;
end std_logic2int;

Assertion
- Überprüfung von Bedingungen und Angaben von errors, warnings
- Meldung wenn Bedingung "false"
- default Meldung: "Assertion violution"

assert <not condition> [report]["<text>"][severity]]serverity_level] { <note|warning|failure>
-default für severity_level ist error

Bsp.:
assert (a = '1' and b = '1')
        report "a and b not equal '1' at same time" severity warning;
```