Generic Példák

A lineáris keresés sablonban

- Implementáljuk a lineáris keresés tételét sablonok segítségével
- Paraméterek: elem, index, tömb típus és a keresett feltétel
- A sablon legyen eljárás
- Out paraméterben jelezze van-e adott tulajdonságú elem és ha van akkor melyik az az elem

Lineáris keresés

```
generic
    type Elem is private;
    type Index is (<>);
    type T is array ( Index range <> ) of Elem;
    with function Prop( A: Elem ) return Boolean;
procedure Linker (x: T; b: out Boolean; j: out Index);
```

Linker

```
procedure Linker (x: T; b: out Boolean; j: out Index) is
begin
    b:= false;
    for i in reverse x'range loop
        if Prop(x(i)) then b:= true; j:= i; end if;
    end loop;
end linker;
```

Linker demo

```
with linker, Ada.Text_IO;
use Ada.Text_IO;
procedure mainlinker is
    type Index is new Integer;
    type Elem is new Integer;
    type T is array (Index range <>) of Elem;
    function myprop (x: Elem) return Boolean is
        begin return (x<0); end myprop;
    k: Index; vane: Boolean;
    a: T(1..5):=(1,2,3,4,5);
    a1: T(1..5):=(1,-2,3,-4,5);
    a2: T(1..5):=(1,2,3,4,-5);
procedure Mylinker is new linker (Elem, Index, T, myprop);
```

Linker demo

```
begin
    mylinker(a, vane, k);
    if vane then Put_Line( Elem'Image(a(k)) );
        else Put_Line("nincs negativ elem"); end if;
   mylinker(a1, vane, k);
   if vane then Put_Line( Elem'Image(a1(k)) );
     else Put_Line("nincs negativ elem"); end if;
   mylinker(a2, vane, k);
   if vane then Put_Line( Elem'Image(a2(k)) );
   else Put_Line("nincs negativ elem"); end if;
end mainlinker;
```

A Felt_Max_ker sablon

- Implementáljuk a feltételes maximum keresés tételét sablonok segítségével
- Paraméterek: elem, index, tömb típus és a keresett feltétel
- A sablon legyen eljárás
- Out paraméterben jelezze van-e adott tulajdonságú elem és ha van akkor melyik az az elem

A sablon paraméterei

```
generic
    type Elem is private;
    type Index is (<>);
    type Tömb is array (Index range <>) of Elem;
    with function Feltétel ( A: Elem ) return Boolean;
    with function "<" ( A, B: Elem ) return Boolean is <>;
procedure Felt_Max_Ker ( T: in Tömb; Van: out Boolean;
                                     Max: out Elem );
```

A sablon eljárás törzse

```
procedure Felt_Max_Ker ( T: in Tömb; Van: out Boolean;
                                     Max: out Elem ) is
   Mh: Index;
begin
   Van := False;
  for I in T'Range loop
     if Feltétel(T(I)) then
          if Van then if T(Mh) < T(I) then Mh := I; end if;
          else Van := True; Mh := I; end if; end if; end loop;
   Max := T(Mh);
end Felt_Max_Ker;
```

Felt_Max_Ker demo

```
with Felt_Max_Ker, Ada.Integer_Text_IO, Ada.Float_Text_IO;
use Ada.Integer_Text_IO, Ada.Float_Text_IO;
procedure Max_Demo is
    type T is array (Integer range <>) of Float;
    function Egész ( A: Float ) return Boolean is
    begin return A = Float(Integer(A)); end Egész;
    procedure Max is new Felt_Max_Ker(Float,Integer,T,Egész);
   A: T(1..10) := (1.4,5.2,3.6,7.0,2.0,65.5,3.0,56.0,2.0,56.0);
   F: Float; V: Boolean;
begin
Max(A,V,F);
if V then Put( F ); end if; end;
```

Map generic

```
generic
    type A is private;
    type B is private;
    type Index is (<>);
    type TA_Array is array ( Index range <> ) of A;
    type TB_Array is array ( Index range <> ) of B;
    with function Op(x: A) return B;
function Map(ta: TA_Array) return TB_Array;
```

Map generic

```
function Map(ta: TA_Array) return TB_Array is
      tb:TB_Array(ta'Range);
begin
for i in ta'Range loop
     tb(i):=op(ta(i));
end loop;
return tb;
end Map;
```

Map demo

```
with map, Ada.Text_IO;
use Ada.Text_IO;
procedure Map_demo is
    type t1 is array (Integer range <>) of Integer;
    type t2 is array (Integer range <>) of Float;
    function square (x: Integer) return Float is
        begin return Float(x*x); end square;
function my_map is new map(Integer, Float, Integer, t1, t2, square);
a: t1(1..5):=(1, 2, 3, 4, 5); b: t2(a'range);

begin b:=my_map(a);
for i in b'Range loop Put_Line(Float'Image(b(i))); end loop;
end Map_demo;
```

Csere a tömbben

```
generic
    type Elem is private;
    type Index is (<>);
    type T is array(Index range <>) of Elem;
procedure reversal (a: in out T);
```

Csere a tömbben

```
procedure reversal (a: in out T) is
   i: Index:= a'First;
   j: Index:= a'Last;
   tmp : Elem;
begin
    while i<j loop
            tmp:=a(i);
            a(i):=a(j);
            a(j):=tmp;
  i:=Index'Succ(i);
j:=Index'Pred(j);
end loop; end reversal;
```

Csere - demo

```
with reversal, Ada.Text_IO; use Ada.Text_IO;
procedure reversalmain is
    type T1 is array (Integer range <>) of Integer;
procedure myreversal is new reversal(Integer, Integer, T1);
a: T1(10..15):=(1,2,3,4,5,6);
a1: T1(10..16):=(1,2,3,4,5,6,7);
a2: T1:=(1,2); a3: T1(1..1); a4: T1(1..0);
begin
myreversal(a);
for i in a'range loop Put_Line(Integer'Image(a(i))); end loop;
end reversalmain;
```

Rendezés – generic-ben generic

- Sablonban másik sablont példányosítunk
- Például a rendezésben használhatjuk a csere sablon eljárást és a max_fely sablon függvényt
- Használat előtt példányosítani kell (még akkor is ha nem ismerjük a típust)

A Csere sablon eljárás

```
generic
    type T is private;
procedure Cserel ( A, B: in out T );

procedure Cserel ( A, B: in out T ) is
    Tmp: T := A;
begin
   A := B;
   B := Tmp;
end Cserel;
```

A Max_Hely sablonfüggvény

```
generic
    type Elem is limited private;
    type Index is (<>);
    type Tömb is array (Index range <>) of Elem;
    with function "<" ( A, B: Elem ) return Boolean is <>;
function Max_Hely ( T: Tömb ) return Index;
```

A sablon függvény törzse

```
function Max_Hely ( T: Tömb ) return Index is
  Mh: Index := T'First;
begin
   for I in T'Range loop
       if T(Mh) < T(I) then Mh := I;
       end if;
   end loop;
   return Mh;
end Max_Hely;
```

Sablonban sablon

```
with Max_Hely, Cserel;
procedure Rendez ( T: in out Tömb ) is
  procedure Cserél_Elem is new Cserel(Elem);
  function Max_Hely_Tömb is new Max_Hely(Elem,Index,Tömb);
    Mh: Index;
   begin
       for I in reverse T'Range loop
       Mh := Max_Hely_Tömb( T(T'First..I) );
      Cserél_Elem( T(I), T(Mh) );
       end loop;
end Rendez;
```

Főprogram

```
with Ada.Text_IO, Rendez; use Ada.Text_IO;
procedure Rendezes is
   type Tömb is array (Character range <>) of Float;
   procedure R_N is new Rendez(Float,Character,Tömb);
   procedure R_Cs is new Rendez(Float,Character,Tömb,">");
   T: Tömb := (3.0,6.2,1.7,5.2,3.9);
begin
   R_Cs(T);
   for I in T'Range loop
       Put_Line( Float'Image( T(I) ) );
   end loop;
end Rendezes;
```

Sablon függvény példa

Készíts sablonfüggvényt `Has_Repetition` névvel, melyet egy indefinit vector nevű tömb típussal (valamint annak index- és elemtípusával) lehet paraméterezni.

A sablonfüggvény példányosításával olyan függvényhez jutunk, amely egy vectort kap paraméterül, és logikai értéket ad vissza: igazat, ha a vector-ban van olyan elem, amely egyenlő a rákövetkezőjével (azaz `v(i) = v(i+1)`).

A főprogramban példányosítod a fenti sablon függvényt.

Teszteld kimerítően a függvényt (szélsőséges esetekre is).

A sablon függvény

```
generic
  type Elem is private;
  type Index is (<>);
  type Vector is array ( Index range <> ) of Elem;
function has_repetition( T: Vector) return Boolean;
```

A sablon függvény törzse

```
function has_repetition( T: Vector) return Boolean is
begin
    if T'length> 1 then
        for i in T'First..Index'Pred(T'Last) loop
            if T(i) = T(Index'Succ(i)) then return True;
            end if;
        end loop;
    end if;
   return False;
end has_repetition;
```

Demo

```
with has_repetition, Ada.Text_IO; use Ada.Text_IO;
procedure demo is
    type TInt is array (Integer range <>) of Integer;
    function my_rep is new has_repetition(Integer, Integer, TInt);
   v1: TInt := (1,1,2,4,5,650);
   v2: TInt := (1,2,3,4,5,6);
   v3: TInt(1..1); --:= (1);
   v4: TInt := (1,2, 3,3,3,56);
   v5: TInt := (1,2, 3,56,56);

begin
v3(1):= 3;
put_line(Boolean'Image(my_rep(v1))); put_line(Boolean'Image(my_rep(v2)));
put_line(Boolean'Image(my_rep(v3))); put_line(Boolean'Image(my_rep(v4)));
put_line(Boolean'Image(my_rep(v5)));
end demo;
```