14. Seznamy prvků

V kpt. 8 jsme se seznámili s datovým typem pole, který umožňuje v programu reprezentovat uspořádanou n-tici prvků. Tu můžeme chápat jako posloupnost n hodnot s konstantním n. V kpt 8. jsme se rovněž naučili v této n-tici vyhledávat největší a nejmenší prvek, rovněž tak měnit uspořádání této n-tice. V programátorské praxi však často pracujeme se strukturou, která reprezentuje konečnou posloupnost s proměnným n – do této posloupnosti je často potřeba prvky přidávat, či naopak z této posloupnosti prvky odebírat. Taková datová struktura se v programátorské praxi nazývá seznam.

Jestliže chceme seznam reprezentovat datovým typem pole, znamená to, že musíme mít představu o maximální délce seznamu (abychom pole mohli deklarovat a aby se do něj v každém okamžiku vešla všechna potřebná data). Tento seznam pak deklarujeme jako pole o této maximální délce. Dále je třeba pracovat s celočíselnou proměnnou, ve které uchováváme aktuální délku seznamu.

1. Příklad: Sestavme program na tvorbu obecného seznamu s možností vkládání prvku na libovolnou pozici, resp. odebírání prvku z libovolné pozice (tato „libovolná pozice“ musí být samozřejmě „uvnitř“ seznamu, tj. minimálně rovna jedné, maximálně aktuální délce seznamu).

```
Const MaxDelka = 20;
  {maximální délka seznamu}
var
  a : array [1..MaxDelka] of byte;
     {vkládané a vybírané hodnoty}
  DelkaSeznamu : Byte;
          {aktuální délka seznamu}

Procedure TForm1.TiskSeznamu;
var i:Byte;
begin
  for i:=1 to MaxDelka do
  begin
  StringGrid1.Cells[0,Pred(i)]:='';
  StringGrid1.Cells[1,Pred(i)]:='';
  end;
  for i:=1 to DelkaSeznamu do
  begin
  StringGrid1.Cells[0,Pred(i)]:=
                        IntToStr(i);
  StringGrid1.Cells[1,Pred(i)]
                   :=IntToStr(a[i]);
  end;
end;
```

Procedura TiskSeznamu je metoda objektu typu TForm1. Tuto proceduru je třeba přidat do deklarace tohoto typu, a to buď do sekce **`private`** nebo **`public`**. Nastavení výchozí délky seznamu na nulu spojíme s událostí **`onCreate`** formuláře:

```
procedure TForm1.Inicializace(Sender: TObject);
begin
  DelkaSeznamu:=0;
end;
```

Následují procedury na vložení prvku na zadanou pozici a vyzvednutí prvku z dané pozice. Potřebné údaje jsou načteny z příslušných SpinEditů:

```
procedure TForm1.Vloz(Sender: TObject);
var i,Pozice:Byte;
    Hodnota :Byte;
begin
 Pozice:=SpinEdit2.Value;
 Hodnota:=SpinEdit1.Value;
 if (Pozice>succ(DelkaSeznamu)) or (DelkaSeznamu=MaxDelka)
    then ShowMessage('Nelze vložit!')
    else begin
          if Pozice<=DelkaSeznamu then
             for i:=DelkaSeznamu downto Pozice do a[succ(i)]:=a[i];
          a[Pozice]:=Hodnota;inc(DelkaSeznamu);
          TiskSeznamu;
         end;
end;

procedure TForm1.Vyjmi(Sender: TObject);
var i,Pozice:Byte;
    Hodnota :Byte;
begin
 Pozice:=SpinEdit3.Value;
 if Pozice>DelkaSeznamu then ShowMessage('Nelze vyjmout!')
    else begin
          Hodnota:=a[Pozice];
          if Pozice<DelkaSeznamu then
             for i:=Pozice to Pred(DelkaSeznamu) do a[i]:=a[succ(i)];
          dec(DelkaSeznamu);
          TiskSeznamu;
         end;
end;
```

Zásadní nevýhodou procedur vkládání či vyzvedávání hodnoty z obecného seznamu je potřeba větší či menší část prvků posunout o jeden index. To je samozřejmě zvlášť u dlouhých seznamů značně nepříjemné. Speciálními typy seznamů, které jsou velmi často používány, jsou zásobník a fronta. Jsou to seznamy, kde vkládáme a odebíráme prvky buď ze začátku či z konce seznamu. Při programování těchto akcí lze posouvání prvků v seznamu obejít.

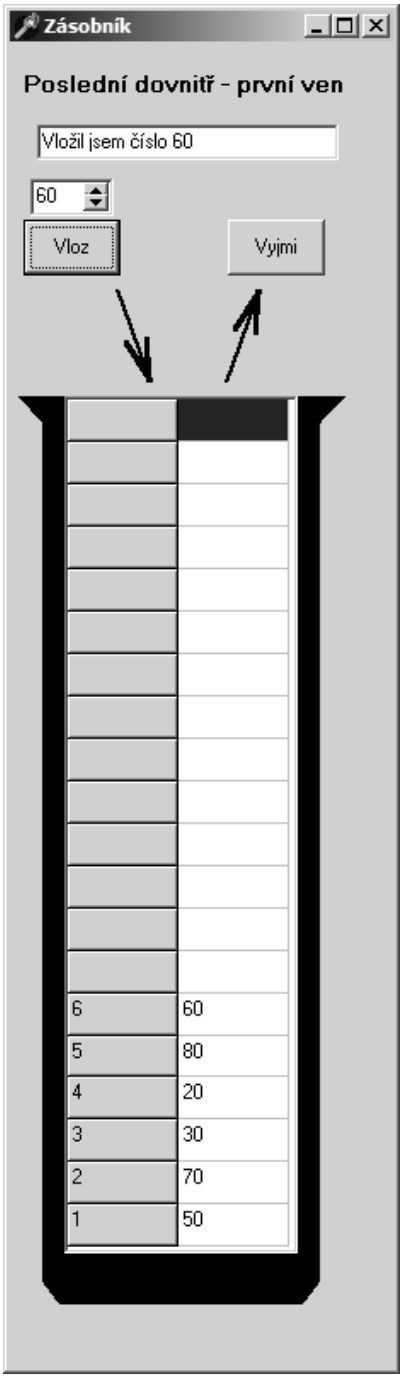
Zásobník je asi nejpoužívanějším typem seznamu. Je to seznam, u něhož jeden konec slouží jak k přidávání, tak k odebírání prvků (tzv. vrchol zásobníku). Prvky nemohou být přidávány či odebírány na druhém konci (dno zásobníku) ani kdekoli jinde. Kdykoli tedy chceme odebrat prvek, bude to ten, který byl do zásobníku vložen jako poslední (je momentálně na vrcholu zásobníku) – zásobník tedy pracuje podle zásady **poslední dovnitř – první ven**. Používá se v případě, kdy je třeba dočasně odložit informace o tom, které dílčí úkoly je třeba ještě vykonat. Zásobník umožňuje zachovat přesně pořadí, ve kterém se mají informace zpracovávat (informace se zpracovávají v opačném pořadí než v jakém přišly na zásobník). Typickým použitím je např. vyhodnocování matematických výrazů. Také samotné volání procedur a funkcí ve vyšších programovacích jazycích využívá techniky zásobníku.

2. Příklad: Sestavne program na přidávání hodnot do zásobníku, resp. na jejich odebírání.

Zásobník nadefinujeme jako záznam, který bude uchovávat jednak **pole vložených hodnot** a jednak **momentální vrchol**:

```
Const Kapacita = 20;
Type TZasobnik = record
                    Prvek : array [1..Kapacita] of Integer;
                    Vrchol: 0..Kapacita;
                 end;
var Zasobnik: TZasobnik;
```

Je-li uživatelské rozhraní navrženo dle připojeného obrazku, pak tisk zásobníku musí respektovat skutečnost, že indexování obkektu StringGrid „jde proti“ indexům prvků zásobníku:




```
Procedure TForm1.TiskniZasobnik;
var i:Integer;
begin
   For i:=0 to Kapacita do
       begin  {vymazání tabulky}
          StringGrid1.Cells[0,i]:='';
          StringGrid1.Cells[1,i]:='';
       end;
   if Zasobnik.Vrchol>0 then
   For i:=1 to Zasobnik.Vrchol do
       begin      {není-li zásobník prázdný}
       StringGrid1.Cells
          [0,Kapacita-Zasobnik.Vrchol+i-1]:=
       IntToStr(Zasobnik.Vrchol-i+1);
       StringGrid1.Cells
          [1,Kapacita-Zasobnik.Vrchol+i-1]:=
       IntToStr(Zasobnik.Prvek
                     [Zasobnik.Vrchol-i+1]);
       end;
end;
```

Nastavení vrcholu prázdného zásobníku spojíme s událostí onCreate formuláře:

```
procedure TForm1.Inicializace(Sender: TObject);
begin
  Zasobnik.Vrchol:=0;
end;
```

Následují procedury na vkládání resp. vybírání:

```
procedure TForm1.Vloz(Sender: TObject);
begin
  With Zasobnik do
  begin
    if Vrchol=Kapacita
    then EditHlaska.Text:=
                'Zásobník zaplněn - nelze vložit'
    else begin
    inc(Vrchol);
    Prvek[Vrchol]:=SpinEdit1.Value;
    EditHlaska.Text:=
    'Vložil jsem číslo '+IntToStr(Prvek[Vrchol]);
         end;
  end;
  TiskniZasobnik;
end;
```

```
procedure TForm1.Vyjmi(Sender: TObject);
begin
  With Zasobnik do
  begin
   if Vrchol=0
      then EditHlaska.Text:='Zásobník je prázdný - nelze vyjmout'
      else begin
            EditHlaska.Text:= 'Vyjmul jsem číslo '+IntToStr(Prvek[Vrchol]);
            dec(Vrchol);
           end;
  end;
  TiskniZasobnik;
end;
```

Fronta: V tomto seznamu se s daty pracuje způsobem, který odpovídá čekání lidí ve frontě v běžném životě. Přicházející lidé (data) se řadí dio fronty, kdo dřív přišel, bude dříve obsloužen a odejde. Do tohoto typu seznamu jsou proto data vkládána jedním koncem („příchodem“) a odebírána druhým koncem („odchodem“). Fronta se tedy řídí zásadou **první dovnitř – první ven**. Také fromta zachovává pořadí, ve kterém se mají informace zpracovávat, na rozdíl pd zásobníku se však informace se zpracovávají ve stejném pořadí, v jakém se řadí do fronty. Typickým použitím je např. prohledávání stromových struktur.

Při konkrétní programové realizaci fronty lze opět použít typ záznam, který bude obsahovat **pole uložených hodnot** a **dva celočíselné typy pro uložení** aktuálního **začátku a konce fronty** (na rozdíl od zásobníku, kde jsme pracovali pouze s jednou hodnotou – vrcholem zásobníku).

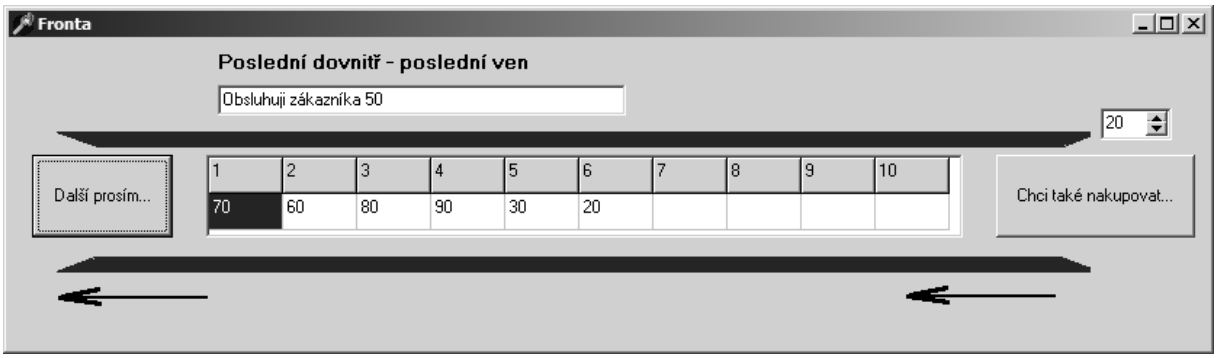
Nechceme-li při vkládání resp. odebírání pole hodnot „fyzicky posouvat“ tak, jako u obecného seznamu – viz př. 1 (což je sice možné, ale krajně neefektivní), je třeba při každém vkládání resp. odebírání posouvat začátek a konec fronty – hodnoty obou proměnných se budou neustále zvyšovat. Tak se ovšem může stát, že i když aktuální délka fronty bude menší než velikost pole, nebude možné vložit další prvek, neboť poslední prvek pole je již obsazen. Tomuto nebezpečí je třeba čelit. Nabízejí se tři základní varianty řešení:

a) Po každém vypuštění prvku posunout prvky pole o index dopředu. Začátek fronty tak bude neustále udržován na prvním indexu. Postup správný, jednoduchý, ovšem (jak již bylo řečeno) značně neefektivní.

b) Indexy v poli posouvat pouze tehdy, až je to nezbytně nutné, tj. v případě, chceme-lui zařadit další prvek a nemáme kam (poslední prvek pole je obsazen). V tom případě přemístíme frontu tak, aby začínala opět prvním indexem. Přesuny hodnot se tak provádějí najednou o větší vzdálenost a podstatně méně často. Výpočet je tak efektivnější.

c) Nejlepší samozřejmě je data nepřesouvat vůbec. Představme si pole modelující frontu nikoli jako body na úsečce, ale body na kružnici. Po „posledním“ indexu n tak vždy „následuje“ index jedna. Jestliže dékla fronty nepřekročí délku pole, je možné vždy vkládat další prvek, aniž abychom museli posouvat prvky v poli. Takto chápaná fronta je nejefektivnější, ponškud komplikovanější je jen kontrola délky fronty (zda je fronta prázdná či naopak zda její dékla nepřekročila délku deklarovaného pole). Pro tento účel je vhodné zavést ještě jednu celočíselnou proměnnou, která bude udávat aktuální délku fronty.

2. Příklad: Sestavne program na přidávání hodnot do fronty, resp. na jejich odebírání, a to podle předchozího bodu c.




```
Const Max=10;
Type TFronta = record
                 Cekajici      : array [1..Max] of Integer;
                 Prvni,Posledni: 1..Max;
                 DelkaFronty   : 0..Max;
               end;
Var  Fronta:TFronta;

procedure TForm1.Init(Sender: TObject);     {onCreate}
var i:Integer;
begin
  With Fronta do
  begin
   Prvni:=1;
   Posledni:=Max;
   DelkaFronty:=0;
  end;
  for i:=1 to Max do StringGrid1.Cells[i-1,0]:=IntToStr(i);
end;

Procedure TForm1.TiskFronty;
var i:Integer;
begin
  With Fronta do
  begin
    for i:=1 to Max do  StringGrid1.Cells[i-1,1]:='';
    for i:=1 to DelkaFronty do
    if Prvni+i-1<=Max
      then StringGrid1.Cells[i-1,1]:= IntToStr(Cekajici[Prvni+i-1])
      else StringGrid1.Cells[i-1,1]:=IntToStr(Cekajici[Prvni+i-Max-1])
  end;
end;

procedure TForm1.Prichod(Sender: TObject);
begin
  With Fronta do
  begin
    if DelkaFronty=Max
      then EditHlaska.Text:='Dalšího zákazníka bohužel nelze přijmout'
      else begin
             if Posledni=Max then Posledni:=1
                             else inc(Posledni); {nový konec fronty}
             Cekajici[Posledni]:=SpinEdit1.Value;
             EditHlaska.Text:='Přišel zákazník '
                                        +IntToStr(Cekajici[Posledni]);
```

```
            inc(DelkaFronty);
          end;
   if DelkaFronty<=Max then TiskFronty;
  end;
end;

procedure TForm1.Odchod(Sender: TObject);
begin
  With Fronta do
  begin
    if DelkaFronty=0
      then EditHlaska.Text:='Už nikdo nečeká, můžete zavřít krám'
      else begin
            EditHlaska.Text:='Obsluhuji zákazníka '
                                              +IntToStr(Cekajici[Prvni]);
            if Prvni=Max then Prvni:=1
                         else inc(Prvni); {nový začátek fronty}
            dec(DelkaFronty);
          end;
     if DelkaFronty<=Max then TiskFronty;
  end;
end;
```

Vyhodnocování matematických výrazů: Typickým příkladem na využití zásobníku je vyhodnocování matematických výrazů. Pro jednoduchost budeme předpokládat aritmetický výraz, tj. výraz, ve kterém se vyskytují jen konstanty, aritmetické operátory +,-,*,/ a kulaté závorky s libovolnou možností vnoření. Z matematiky i z psaní programů jsme zvyklí na běžný zápis takového výrazu do řádku. Každý binární operátor je umístěn mezi dvěma operandy, s nimiž má být operace provedena. Jejich priorita je určena především závorkami. Operátory na stejné závorkové úrovni se vyhodnocují dle definované priority (násobení a dělení má vyšší prioritu než sečítání a odčítání), operátory stejné priority pak zleva doprava. Tento zápis se běžně nazývá infixový – např. (65-3*5)/(2+3). Pro člověka je tento zápis relativně přehledný a snadno vyhodnotitelný. Pro strojové zpracování jsou výhodnější zápisy jiné.

Postfixová notace: Operátor se píše za dva operandy, k nimž se vztahuje: 65 3 5 *-2 3 + /
Prefixová notace: Operátor se píše před dva operandy, k nimž se vztahuje: / - 65 * 3 5+2 3
(všimněte si, že tyto zápisy nepotřebují závorky).

3. Příklad: Vyhodnocování aritmetického výrazu zapsaného v běžné infixové notaci. Provedeme přes převod na postfixovou notaci. Použijeme funkci prvek, která čte vstupní řetězec znak po znaku a vraci vrací true a znak, pokud se četl znak nebo false, hodnotu a znak "$" pokud se četlo číslo:

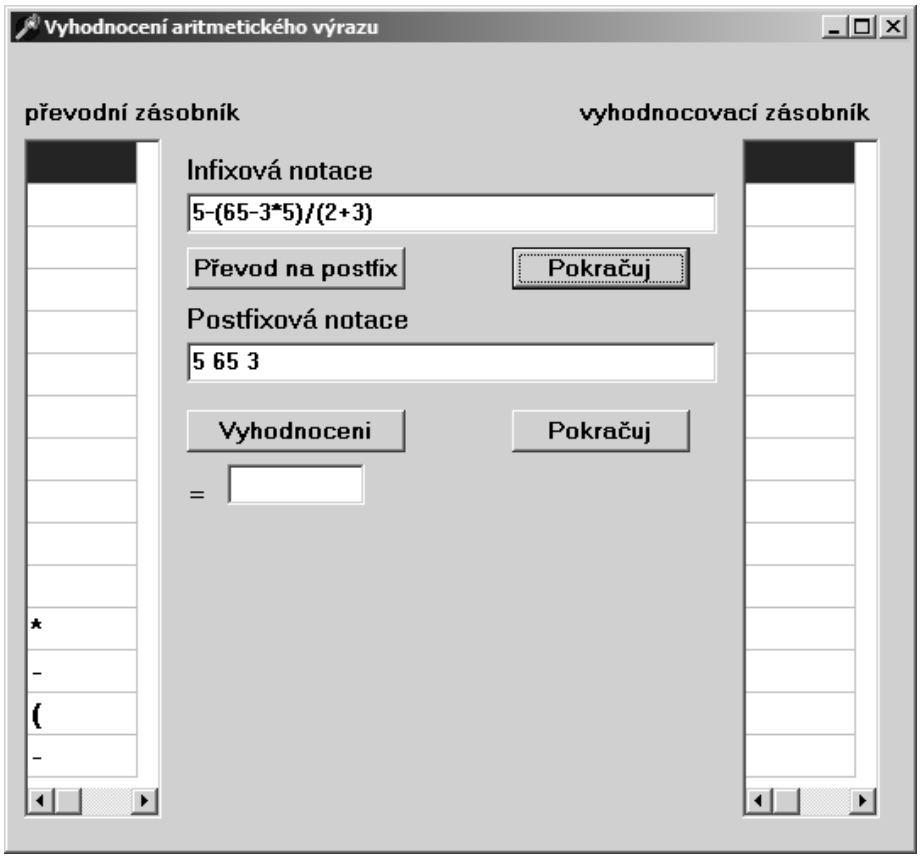
```
Function Prvek
       (Zapis:String;i:Integer; var Hodnota:Double; var Znak:Char):Boolean;
var j      :Integer;
    Cislice:Set of Char;
begin
    Cislice:=['0'..'9','.'];
    if i>Length(zapis)
          then Prvek:=False
          else begin
                 Prvek:=True;
                 if Zapis[i] in cislice
                    then begin
                           Znak:='$'; {na vstupu je číslo}
```

```
                        j:=i;
                        Repeat
                          inc(i);
                        Until not(Zapis[i] in cislice);
                        Hodnota:=StrToFloat(Copy(Zapis,j,i-j));
                      end
                 else Znak:=Zapis[i]; {Precte jiny znak}
            end;
end;
```

Při převodu na postfix se pomocí této funkce čte vstupní řetězec. Pokud se čte číslo, naplňuje se výstupní postfixová notace. Operátory a levé závorky se ukládají do zásobníku. Přečte-li se pravá závorka, vyprázdní se postupně zásobník až po levou závorku, která se rovněž ze zásobníku vyjme (do postfixu se však závorky nezapisují). Přečte-li se celý vstupní řetězec a je-li zapsán korektně, pak v zásobníku zbývají pouze operátory, které je třeba postupněš vyjmout a připsat do postfixu:

```
Procedure InfixNaPostfix(Vstup:String;var Vystup:String);
Const Max = 15 ;                                  {maximalni pocet operatoru}
var   Zasobnik  : array [0..Max] of Char;              {Pracovni zasobnik}
      W         : 0..Max;                              {Vrchol zasobniku}
      H         : Double;                              {hodnoty operandů}
      Z         : Char;                                  {znak na vstupu}
      Pokracovat: Boolean;
      i:          Integer;
begin
  w:=0;i:=1;
  Pokracovat:=Prvek(Vstup,i,H,Z);Vystup:='';
  While Pokracovat do
  begin
   Case Z of
   '$':begin Vystup:=Vystup+FloatToStr(H)+' ';         {na vstupu je cislo}
             i:=i+Length(FloatToStr(H));
       end;
   '(':begin inc(W);Zasobnik[w]:='(';inc(i);end;     {zavorku do zasobniku}
   ')':begin                              {z vrcholu zasobniku az k zavorce}
         while (w>0) and (Zasobnik[w]<>'(') do
           begin
             Vystup:=Vystup+Zasobnik[w];dec(w);
           end;
           if w=0 then Vystup:='Chybny vyraz'
                  else dec(w);                          {zruseni zavorky}
         inc(i);
       end;
 '/','*':begin             {operatory nejvyssi priority - ven ze zasobniku}
           while (w>0) and (Zasobnik[w] in ['*','/']) do
             begin
                 Vystup:=Vystup+Zasobnik[w];dec(w);
             end;
             inc(w);Zasobnik[w]:=Z;                    {znak do zasobniku}
           inc(i);
         end;
 '+','-':begin               {operatory nizsi priority - ven ze zasobniku}
           while (w>0) and (Zasobnik[w] in ['+','-','*','/']) do
             begin
                 Vystup:=Vystup+Zasobnik[w];dec(w);
             end;
             inc(w);Zasobnik[w]:=Z;                    {znak do zasobniku}
           inc(i);
         end;
```

```
       end;                                                    {Case}
      Pokracovat:=Prvek(Vstup,i,H,Z);
  end;
  While (w>0) and (Zasobnik[w]<>'(') do
       begin                        {vyprazdnit pripadny zbytek zasobniku}
        Vystup:=Vystup+Zasobnik[w];dec(w);
       end;
  if w>0 then Vystup:='Chybny vyraz';
end;                                          {Procedure InfixNaPostFix}
```




```
procedure TForm1.Vyhodnoceni_Postfixu(Sender: TObject);
Const Max = 15;                                  {maximalni délka řetězce}
var   Zasobnik  : array [0..Max] of Double;          {zásobník na hodnoty}
      i,w       : Integer;
      H,H1,H2   :Double;
      Vstup, VypisCisla : String;
      Cislice   : Set Of Char;
      Pokracovat:Boolean;
      Z         :Char;
begin
  Vstup:=EditPostFix.Text;
  Cislice:=['0'..'9','.'];
  for i:=0 to Max do Zasobnik[i]:=0;
  w:=-1;i:=1;
  Pokracovat:=Prvek(Vstup,i,H,Z);
  While Pokracovat do
  begin
    Case Z of
   '$':begin inc(w);                                  {na vstupu je cislo}
             i:=i+succ(Length(FloatToStr(H)));
             Zasobnik[w]:=h;
       end;
```

```
  '/','*', '+','-':
        begin
          H1:=Zasobnik[w];
          dec(w);
          H2:=Zasobnik[w];
           Case Z of
            '/':Zasobnik[w]:=H2 /   H1;
            '*':Zasobnik[w]:=H2 *   H1;
            '+':Zasobnik[w]:=H2 +   H1;
            '-':Zasobnik[w]:=H2 -   H1;
           end;
        inc(i);
        end;
      end;                                                        {Case Z}
      Pokracovat:=Prvek(Vstup,i,H,Z);
   end;
  While (w>0) do  dec(w);            {vyprazdnit pripadny zbytek zasobniku}
  Edit3.Text:=FloatToStr(Zasobnik[w]);
end;
```