Halmaz-típuskonstrukció
elemi (felsorolás-) típusú elemekből

A feladat

A „magyar kártya” típusának megvalósítása.
Tisztázandók:
1. Milyen *műveletek* „értelmesek” a típussal kapcsolatosan?
2. Hogyan *ábrázolható* a típus?
3. Miként *valósíthatók meg a műveletek* figyelembe véve a reprezentációt?

Szétválasztottuk a feladatban található 4 típus-fogalmat egy-egy unitba: egy Szin (⇒TSzin), egy Figura (⇒TFigura) és egy Kartya (⇒TKartya) típust kódoló unitra, továbbá egy Pakli (⇒TPakli) *típuskonstrukció*t kódoló unitra.

A főprogram

Elvárásomat lásd futás közben: PakliPr.exe.
Az alábbi főprogram forráskódja: PakliPr.pas.

```
Program HalmazPr;
(*
   Bitvektoros halmazábrázolás példája. Főprogram.
*)
  Uses
    Crt,PakliUn,KartyaUn; {PakKeKe=a keret; PakliUn=a kész}
  {$i AltRutin.inc}
  Type
    TKartyak = Array [1..MaxKartyaDb] of TKartya;
  Var
    asztal,kez:THalmaz;
    k:Tkartya;
    kevertLapok:TKartyak;

Begin
  UjLap('Kártyás program',0);
  Writeln;
  TeljesPakli(asztal); KiPakli('Asztal:',asztal);
  Writeln;
  UresPakli(kez); KiPakli('Kéz:',kez);
  Writeln;
  While not UresPakliE(asztal) do
  Begin
    VeletlenLap(asztal,k); KartyaUn.Ki('Kihúzott kártya:',k,CrLf);
    Elvesz(asztal,k); KiPakli('Asztal:',asztal);
    Kozetesz(kez,k); KiPakli('Kéz:',kez);
    LegjobbLap(kez,k); KartyaUn.Ki('Legjobb lapom:',k,CrLf);
    LegrosszabbLap(kez,k); KartyaUn.Ki('Legrosszabb lapom:',k,CrLf);
    Writeln;
    BillreVar;
  End;
  UjLap('Kártyakever,s',-1);
  {Kártyakeverés a kevertLapok tömbbe:}
  {... hf ...}
  BillreVar;
End.
```

A típusok megvalósítása Pascal unitok-tal

Az alábbi unit (a típus „keretének” a) forráskódja: PakKeUn.pas.

```
Unit PakKeUn;
(*
   Bitvektoros halmazábrázolás példája.
   Magyar kártya-halmaz típusa.
   Asszociált müveletek:      (Készség)
   -------------- THalmaz(TKartya) ---------------
   * TeljesPakli                  +
   * UresPakli                    -
   * TeljesPakliE                 -
   * UresPakliE                   +
   * Kozetesz                     +
   * Elvesz                       +
   * BenneVanE                    -
   * Egyesit                      -
   * Kozosek                      -
   * LegjobbLap                   +
   * LegrosszabbLap               -
   * VeletlenLap                  +
   . . . . . .
   * BePakli                      -
   * KiPakli                      +
   * HibasEPakli                  +
*)
  Interface
    Uses
      KartyaUn,SzinUnit,FiguUnit;
    Type
      THalmaz = Record
                  lapok: Array [TSzin,TFigura] of Boolean;
                  siker: Boolean;
                End;
    Const
      MaxKartyaDb = (Ord(MaxFigura)+1)*(Ord(MaxSzin)+1);

    Procedure TeljesPakli(Var p:THalmaz);
    {Ef: -
     Uf: MINDEN s:TSzin, MINDEN f:TFigura: p.lapok[s,f] ES
         p.siker}
    Function TeljesPakliE(Const p:THalmaz):Boolean;
    {Ef: -
     Uf: TeljesPakliE(p) <=> MINDEN s:TSzin, MINDEN f:TFigura:
                                                    p.lapok[s,f]}
    Procedure UresPakli(Var p:THalmaz);
    {Ef: -
     Uf: MINDEN s:TSzin, MINDEN f:TFigura: NEM p.lapok[s,f] ES
         p.siker}
    Function UresPakliE(Var p:THalmaz):Boolean;
    {Ef: -
     Uf: UresPakliE(p) <=> MINDEN s:TSzin, MINDEN f:TFigura:
                                                NEM p.lapok[s,f]}
    Procedure Kozetesz(Var p:THalmaz; Const k:TKartya);
    {Ef: -
     Uf: k=(s,f) ES p.lapok[s,f]}
    Procedure Elvesz(Var p:THalmaz; Const k:TKartya);
    {Ef: k=(s,f) ES p.lapok[s,f]
     Uf: k=(s,f) ES NEM p.lapok[s,f]}
```

```
  Function  BenneVanE(Var p:THalmaz; Const k:TKartya):Boolean;
  {Ef: -
   Uf: BenneVanE(p,k) <=> k=(s,f) ES p.lapok[s,f] }
  Procedure Egyesit(Const p1,p2:THalmaz; Var p:THalmaz);
  {Ef: -
   Uf: ...}
  Procedure Kozosek(Const p1,p2:THalmaz; Var p:THalmaz);
  {Ef: -
   Uf: ...}
  Procedure LegjobbLap(Var p:THalmaz; Var k:TKartya);
  {Ef: NEM UresPakliE(p)
   Uf: k=(s,f) ES MINDEN kk:TKartya: kk=(ss,ff) ES
                                     BenneVanE(ss,ff) ES
                                     JobbE((ss,ff),(s,f))}
  Procedure LegrosszabbLap(Var p:THalmaz; Var k:TKartya);
  {Ef: NEM UresPakliE(p)
   Uf: ...}
  Procedure VeletlenLap(Var p:THalmaz; Var k:TKartya);
  {Ef: NEM UresPakliE(p)
   Uf: BenneVan(p,k)}
  Procedure BePakli(Const kerd:String; Var p:THalmaz);
  {Ef: -
   Uf: ...}
  Procedure KiPakli(Const szov:String; Const p:THalmaz);
  {Ef: -
   Uf: ...}
  Function HibasEPakli(Var p:THalmaz):Boolean;
  {Ef: -
   Uf: HibasEPakli=p.siker ES p'.siker}

Implementation

  Uses {Newdelay,}Crt; {az AltRutin.inc kívánja meg}
  {$i AltRutin.inc}

  {------------------------ KártyaHalmaz: -------------------------}

  Procedure TeljesPakli(Var p:THalmaz);
  {Ef: -
   Uf: MINDEN s:TSzin, MINDEN f:TFigura: p.lapok[s,f] ES
       p.siker}
    Var
      s:TSzin; f:TFigura;
  Begin
    For s:=MinSzin to MaxSzin do
    Begin
      For f:=MinFigura to MaxFigura do
      Begin
        p.lapok[s,f]:=True
      End;
    End;
    p.siker:=True
  End;{TeljesPakli}
```

```
Function TeljesPakliE(Const p:THalmaz):Boolean;
{Ef: -
 Uf: TeljesPakliE(p) <=> MINDEN s:TSzin, MINDEN f:TFigura:
                                                    p.lapok[s,f]}
  Var
    s:TSzin; f:TFigura; db:Byte;
Begin
  {... hf ...}
End;{TeljesPakliE}

Procedure UresPakli(Var p:THalmaz);
{Ef: -
 Uf: MINDEN s:TSzin, MINDEN f:TFigura: NEM p.lapok[s,f] ES
     p.siker}
  Var
    s:TSzin; f:TFigura;
Begin
  {... hf ...}
End;{UresPakli}

Function UresPakliE(Var p:THalmaz):Boolean;
{Ef: -
 Uf: UresPakliE(p) <=> MINDEN s:TSzin, MINDEN f:TFigura:
                                                NEM p.lapok[s,f]}
  Var
    s:TSzin; f:TFigura; db:Byte;
Begin
  db:=0;
  For s:=MinSzin to MaxSzin do
  Begin
    For f:=MinFigura to MaxFigura do
    Begin
      If p.lapok[s,f] then Inc(db)
    End;
  End;
  UresPakliE:=db=0
End;{UresPakliE}

Procedure Kozetesz(Var p:THalmaz; Const k:TKartya);
{Ef: -
 Uf: k=(s,f) ES p.lapok[s,f]}
Begin
  p.lapok[k.szin,k.figura]:=True
End;{Kozetesz}

Procedure Elvesz(Var p:THalmaz; Const k:TKartya);
{Ef: k=(s,f) ES p.lapok[s,f]
 Uf: k=(s,f) ES NEM p.lapok[s,f]}
Begin
  If not p.lapok[k.szin,k.figura] then {nem teljesül az Ef}
  Begin
    p.siker:=False
  End
    Else
  Begin
    p.lapok[k.szin,k.figura]:=False
  End;
End;{Elvesz}
```

```
Function  BenneVanE(Var p:THalmaz; Const k:TKartya):Boolean;
{Ef: -
 Uf: BenneVanE(p,k) <=> k=(s,f) ES p.lapok[s,f]}
Begin
  {... hf ...}
End;{BenneVanE}

Procedure Egyesit(Const p1,p2:THalmaz; Var p:THalmaz);
{Ef: -
 Uf: ...}
Begin
  {... hf ...}
End;{Egyesit}

Procedure Kozosek(Const p1,p2:THalmaz; Var p:THalmaz);
{Ef: -
 Uf: ...}
Begin
  {... hf ...}
End;{Kozosek}

Procedure LegjobbLap(Var p:THalmaz; Var k:TKartya);
{Ef: NEM UresPakliE(p)
 Uf: ...}
  Var
    s:TSzin; f:TFigura;
Begin
  If UresPakliE(p) then {nem teljesül az Ef}
  Begin
    p.siker:=False
  End
    Else
  Begin
    s:=Zold; f:=Asz;
    While ((s>Makk) or (s=Makk) and (f>VII)) and
          not p.lapok[s,f] do
    Begin
      If f>VII then f:=pred(f)
               else Begin s:=pred(s); f:=Asz end;
    End;
    k.szin:=s; k.figura:=f;
  End;
End;{LegjobbLap}

Procedure LegrosszabbLap(Var p:THalmaz; Var k:TKartya);
{Ef: NEM UresPakliE(p)
 Uf: k=(s,f) ES MINDEN kk:TKartya: kk=(ss,ff) ES
                                   BenneVanE(ss,ff) ES
                                   RosszabbE((ss,ff),(s,f))}
  Var
    s:TSzin; f:TFigura;
Begin
  If UresPakliE(p) then {nem teljesül az Ef}
  Begin
    p.siker:=False
  End
    Else
  Begin
    {... hf ...}
  End;
End;{LegrosszabbLap}
```

```
  Procedure VeletlenLap(Var p:THalmaz; Var k:TKartya);
  {Ef: NEM UresPakliE(p)
   Uf: BenneVan(p,k)}
    Var
      s:TSzin; f:TFigura;
  Begin
    If UresPakliE(p) then {nem teljesül az Ef}
    Begin
      p.siker:=False
    End
      Else
    Begin
      Repeat
        s:=Kod2Szin(Random(4)); f:=Kod2Figura(Random(8));
      Until p.lapok[s,f];
      k.szin:=s; k.figura:=f
    End;
  End;{VeletlenLap}

  Procedure BePakli(Const kerd:String; Var p:THalmaz);
  {Ef: ...
   Uf: ...}
  Begin
    {... hf ...}
  End;{BePakli}

  Procedure KiPakli(Const szov:String; Const p:THalmaz);
  {Ef: -
   Uf: ...}
    Var
      s:TSzin; f:TFigura;
  Begin
    Writeln(szov);
    For s:=MinSzin to MaxSzin do
    Begin
      Write(Szin2String(s):6,':');
      For f:=MinFigura to MaxFigura do
      Begin
        If p.lapok[s,f] then Write(Figura2String(f),' ')
      End;
      Writeln;
    End;
  End;{KiPakli}

  Function HibasEPakli(Var p:THalmaz):Boolean;
  {Ef: -
   Uf: HibasEPakli=p.siker ES p'.siker }
  Begin
    HibasEPakli:=p.siker;
    p.siker:=True
  End;{HibasEPakli}

Begin

End.
```

A kártya típus keretéül ezt használhatja (KartyaUn.pas):

```
Unit KartyaUn;
(*
   A magyar kártya típusa.
*)
 Interface

     Uses
       SzinUnit,FiguUnit;
     Type
       TKartya=Record szin:TSzin; figura:TFigura; hiba:Boolean End;
     Procedure Kartya(Const s:TSzin; Const f:TFigura; Var kartyaKi:TKartya);
     Function KartyaSzin(Const k:TKartya):TSzin;
     Function KartyaFigura(Const k:TKartya):TFigura;
     Procedure Be(Const kerd:String; Var k:TKartya);
      {Ef: ... SZIN + ',' FIGURA alakban
       Uf: ...}
     Procedure  Ki(Const szov:String; Const k:TKartya; Const uto:String);
      {Ef: -
       Uf: ...}
     Function HibasEKartya(Var k:TKartya):Boolean;
      {Ef: -
       Uf: k.hiba     => HibasEKartya=Igaz ES k=TKartya(Min'TSzin,Min'TFigura)
           nem k.hiba => HibasEKartya=Hamis}

 Implementation

    Uses
      {Newdelay,}Crt; {az AltRutin.inc kívánja meg}
    {$i AltRutin.inc}

     Procedure Kartya(Const s:TSzin; Const f:TFigura; Var kartyaKi:TKartya);
      {Ef:
       Uf:}
     Begin
       kartyaKi.szin:=s; kartyaKi.figura:=f
     End;

     Function KartyaSzin(Const k:TKartya):TSzin;
      {Ef: -
       Uf:}
     Begin
       KartyaSzin:=k.szin
     End;

     Function KartyaFigura(Const k:TKartya):TFigura;
      {Ef: -
       Uf:}
     Begin
       KartyaFigura:=k.figura
     End;

     Procedure  Be(Const kerd:String; Var k:TKartya);
      {Ef: ... SZIN + ',' FIGURA alakban
       Uf: ...}
       Var
         sk,ss,sf:String;
         s:TSzin; f:TFigura;
         hol:Integer;
```

```
      Begin {az Ef-et nem ellenörizzük}
        Write(kerd);
        (* egyszerübb lenne:
           SzinUnit.Be('',k.szin); FiguUnit.Be('',k.figura)
        *)
        Readln(sk); sk:=StringUpCase(sk);
        hol:=Pos(',',sk); ss:=Copy(sk,1,hol-1); sf:=Copy(sk,hol+1,Length(sk));
        {Szin-kódolás:}
        k.szin:=String2Szin(ss);
        {Figura-kódolás:}
        k.figura:=String2Figura(sf);
      End;{BeKartya}

    Procedure  Ki(Const szov:String; Const k:TKartya; Const uto:String);
     {Ef: -
      Uf: ...}
    Begin
      Write(szov);
      SzinUnit.Ki('',k.szin,''); FiguUnit.Ki(' ',k.figura,'');
      Write(uto);
    End;

    Function HibasEKartya(Var k:TKartya):Boolean;
     {Ef: -
      Uf: k.hiba     => HibasEKartya=Igaz ES k=TKartya(Min'TSzin,Min'TFigura)
          nem k.hiba => HibasEKartya=Hamis}
    Begin
      If k.hiba then
      Begin
        HibasEKartya:=True; k.szin:=Low(TSzin); k.figura:=Low(TFigura);
      End
        else
      Begin
        HibasEKartya:=False
      End
    End;

Begin

End.
```

A figura típusának félig kész kerete az alábbi (FiguKeUn.pas):

```
unit FiguKeUn;
(*
   Asszociált müveletek:      (Készség)
   -------------- TFigure ---------------
   Be                              -
   Ki                              +
   String2Figura                   +
   Kod2Figura                      +
   Figura2String                   -
   Figura2Kod                      -
   Kov                             -
   Elo                             -
   Kisebb                          -
   HibasESzin                      +
*)
```

```
Interface
  Type
    TFigura=(VII,VIII,IX,X,Also,Felso,kiraly,Asz,NemFigura);
  Const
    MinFigura=Low(TFigura); MaxFigura=Pred(High(TFigura));

  Procedure Be(Const kerd:String; Var f:TFigura);
  Procedure Ki(Const szov:String; Const f:TFigura; Const uto:String);
  Function String2Figura(Const sz:String):TFigura;
  Function Figura2String(Const f:TFigura):String;
  Function Figura2Kod(Const f:TFigura):Integer;
  Function Kod2Figura(Const ko:Integer):TFigura;
  Function Kov(Const f:TFigura):TFigura;
  Function Elo(Const f:TFigura):TFigura;
  Function Kisebb(Const f1,f2:TFigura):Boolean;
  Function HibasEFigura(Var f:TFigura):Boolean;

Implementation
  Const
    SFigura:Array [TFigura] of String=
             ('VII','VIII','IX','X','Also',
              'Felso','Kiraly','Asz','Nem figura');

  Procedure Be(Const kerd:String; Var f:TFigura);
    Var
      fS:String;
  Begin
    {... hf ...}
  End{Be};

  Procedure Ki(Const szov:String; Const f:TFigura; Const uto:String);
  Begin
    Write(szov+Figura2String(f)+uto)
  End{Ki};

  Function String2Figura(Const sz:String):TFigura;
    Var
      i:TFigura;
  Begin
    i:=Low(TFigura);
    While (i<High(TFigura)) and (Figura2String(i)<>sz) do i:=Succ(i);
    String2Figura:=i
  End{String2Figura};

  Function Figura2String(Const f:TFigura):String;
  Begin
    {... hf ...}
  End{Figura2String};

  Function Figura2Kod(Const f:TFigura):Integer;
   {Ef: -
    Uf: Figura2Kod(f)=Sorszam(f)}
  Begin
    {... hf ...}
  End;

  Function Kod2Figura(Const ko:Integer):TFigura;
   {Ef: ko ELEME [0..7]
    Uf: KodbolFigura(ko)=TFigura(ko)}
```

```
  Begin
    Kod2Figura:=TFigura(ko)
  End;

  Function Kov(Const f:TFigura):TFigura;
  Begin
    {... hf ...}
  End{Kov};

  Function Elo(Const f:TFigura):TFigura;
  Begin
    {... hf ...}
  End{Elo};

  Function Kisebb(Const f1,f2:TFigura):Boolean;
  Begin
    {... hf ...}
  End{Kisebb};

  Function HibasEFigura(Var f:TFigura):Boolean;
   {Ef: -
    Uf: f=NemFigura  => HibasEFigura=Igaz ES f=Min'TFigura
        f<>NemFigura => HibasEFigura=Hamis}
  Begin
    If f=NemFigura then
    Begin
      HibasEFigura:=True; f:=Low(TFigura)
    End
      else
    Begin
      HibasEFigura:=False
    End;
  End{HibasEFigura};

Begin

End.
```

A szín típusának félig kész kerete az alábbi (SzinKeUn.pas):

```
unit SzinKeUn;
(*
   Asszociált müveletek:      (Készség)
   -------------- TSzin ---------------
   Be                              +
   Ki                              +
   String2Szin                     -
   Szin2String                     +
   Kod2Szin                        +
   Szin2Kod                        +
   Kov                             -
   Elo                             -
   Kisebb                          -
   HibasESzin                      +
*)
Interface
  Type
    TSzin=(Makk,Piros,Tok,Zold,NemSzin);
  Const
    MinSzin=Low(TSzin); MaxSzin=Pred(High(TSzin));
```

```
  Procedure Be(Const kerd:String; Var sz:TSzin);
  Procedure Ki(Const szov:String; Const sz:TSzin; Const uto:String);
  Function String2Szin(Const sz:String):TSzin;
  Function Szin2String(Const sz:TSzin):String;
  Function Kod2Szin(Const ko:Integer):TSzin;
  Function Szin2Kod(Const s:TSzin):Integer;
  Function Kov(Const sz:TSzin):TSzin;
  Function Elo(Const sz:TSzin):TSzin;
  Function Kisebb(Const sz1,sz2:TSzin):Boolean;
  Function HibasESzin(Var sz:TSzin):Boolean;

Implementation
  Const
    SSzin:Array [TSzin] of String=
          ('Makk','Piros','Tok','Zold','Nem szin');

  Procedure Be(Const kerd:String; Var sz:TSzin);
    Var
      sS:String;
  Begin
    Write(kerd); Readln(sS);
    sz:=String2Szin(sS);
  End{Be};

  Procedure Ki(Const szov:String; Const sz:TSzin; Const uto:String);
  Begin
    Write(szov+Szin2String(sz)+uto)
  End{Ki};

  Function String2Szin(Const sz:String):TSzin;
    Var
      i:TSzin;
  Begin
    {... hf ...}
  End{String2Szin};

  Function Szin2String(Const sz:TSzin):String;
  Begin
    Szin2String:=SSzin[sz]
  End{Szin2String};

  Function Kod2Szin(Const ko:Integer):TSzin;
   {Ef: ko ELEME [0..3]
    Uf: KodbolSzin(ko)=TSzin(ko)}
  Begin
    Kod2Szin:=TSzin(ko)
  End;

  Function Szin2Kod(Const s:TSzin):Integer;
   {Ef: -
    Uf: Szin2Kod(s)=Sorszam(s)}
  Begin
    Szin2Kod:=Ord(s)
  End;

  Function Kov(Const sz:TSzin):TSzin;
  Begin
    {... hf ...}
  End{Kov};
```

```
  Function Elo(Const sz:TSzin):TSzin;
  Begin
    {... hf ...}
  End{Elo};

  Function Kisebb(Const sz1,sz2:TSzin):Boolean;
  Begin
    {... hf ...}
  End{Kisebb};

  Function HibasESzin(Var sz:TSzin):Boolean;
   {Ef: -
    Uf: f=NemSzin  => HibasESzin=Igaz ES f=Min'TSzin
        f<>NemSzin => HibasESzin=Hamis}
  Begin
    If sz=NemSzin then
    Begin
      HibasESzin:=True; sz:=Low(TSzin)
    End
      else
    Begin
      HibasESzin:=False
    End;
  End{HibasESzin};

Begin

End.
```

Az egész anyag összecsomagolva: Pakli Gyak.zip.