1. Zbiory – realizacja tablicowa

```
 { Zadanie.  ------- Utworzyc abstrakcyjny typ Zbior wraz
  z procedurami wykonywanymi na danych typu Zbior. Do
reprezentowania
  zbiorow uzyc tablic. Utworzyc biblioteke Pascala (UNIT)
zawierajaca
  wszystkie napisane definicje i procedury.  Przedyskutowa/c
marno/s/c
  takiej realizacji.  Zastanowi/c si/e jak mo/zna to poprawi/c? }


{ Realizacja 'zbiorow' z zastosowaniem tablic. Przyklad.

  Stale: Moc = ilosc elementow w zbiorze
  type ElementZbioru = liczba czlkowita z zakresu 1..Moc
       Zbior = tablica [1..Moc] o elementach TRUE lub FALSE

  Procedury i funkcje:

           UtworzZbiorUniwersalny(Z)  - proc. (TRUE -> Z[I]
           DodajElement[DoZbioru](E,Z) - proc.
           UsunElement[ZeZbioru](E,Z) - proc.
           UtworzPusty(Z) - FALSE -> Z[I]
           PustyZbior - funkcja; boolean
           RowneZbiory - ,,
           WZbiorze - ,,; sprawdza czy el. nalezy do zbioru
           SumaZbiorow - suma topol. 2 zbiorow
           RoznicaZbiorow - roznica topol. 2 zb.
           IloczynZbiorow - przeciecie 2 zbiorow
           PodzbiorZbioru - funk., boolean

           CzytajZb - czyta niezerowe elementy zbioru z konsoli
           WypiszZb - wypisuje numery niezerowych elementow
}
```

```
uses
   Crt;
const
   Moc = 100;
type
   ElementZbioru = 1..Moc;
   Zbior = array [ElementZbioru] of boolean;


procedure UtworzPusty(var Z: Zbior);
(* oproznia zbior S *)
var
   I: integer;
begin
   for I:=1 to Moc do Z[i]:=false
end;
```

```
procedure UtworzZbiorUniwersalny(var Z: Zbior);
(* wstawia TRUE do Z dla kazdego i*)
var
  I: integer;
begin
   for i:=1 to Moc do Z[i]:=true;
end;



procedure DodajElement(E: ElementZbioru; var Z: Zbior);
(* Dodaje element do zbioru *)
begin Z[E]:=true end;



procedure UsunElement(E: ElementZbioru; var Z: Zbior);
(* Usuwa element zbioru *)
begin Z[E]:=false end;


{
 Funkcja PustyZbior musi sprawdza/c elementy tablicy
 a/z do chwili gdy znajdzie TRUE lub przebiegnie wszystkie
 elementy. U/zyjemy konstrukcji VAR aby zabezpieczy/c si/e
 przed kopiowaniem tablicy, jest to strata
 czasu przy du/zych zbiorach
}

function PustyZbior(var Z: Zbior): boolean;
(* TRUE gdy Z jest pusty *)
var
   I: integer;
begin
{
  powtarzaj az znajdziesz element lub miniesz koniec
}
   I:=0; PustyZbior:=true;
   repeat I:=I+1 until Z[I] or (I>Moc);
   if(I<Moc) then PustyZbior:=false;
end;



function CzyWZbiorze(E: ElementZbioru; Z: Zbior): boolean;
(* zwraca TRUE gdy E nalezy do Z lub
         FALSE gdy E nie nalezy do Z  *)
begin CzyWZbiorze:=Z[E] end;



function RowneZbiory(var Z1, Z2: Zbior): boolean;
(*
 zwraca TRUE gdy Z1=Z2 lub FALSE gdy Z1<>Z2
```

```
*)
var
  I: integer;
begin
{
  powtarzaj az do chwili gdy znajdziesz
  rozne elementy lub dojdziesz do konca
}
  I:=0; RowneZbiory:=false;
  repeat i:=I+1 until (Z1[I] <> Z2[I]) or (I>Moc);
  if (I>Moc) then RowneZbiory:=true;
end;



procedure SumaZbiorow(var Z1, Z2, Suma: Zbior);
{
  Zwraca zbior Suma rowny sumie mnog. Z1 i Z2
}
var
   I: integer;
begin
   for i:=1 to Moc do Suma[I] := Z1[I] or Z2[I];
end;



procedure RoznicaZbiorow(var Z1, Z2, Roznica: Zbior);
{
  Zwraca zbior Roznica rowny roznicy mnog. Z1 i Z2
}
var
   I: integer;
begin
   for i:=1 to Moc do Roznica[I] := Z1[I] and not Z2[I];
end;



procedure IloczynZbiorow(var Z1, Z2, Iloczyn: Zbior);
{
  Zwraca zbior Iloczyn rowny przecieciu Z1 i Z2
}
var
   I: integer;
begin
   for i:=1 to Moc do Iloczyn[I] := Z1[I] and Z2[I];
end;



function Podzbior(var Z1, Z2: Zbior): boolean;
(*  Zwraca TRUE  jesli zbior Z1 jest podzbiorem zbioru Z2
    i      FALSE jesli nie jest *)
```

```
var
   I: integer;
begin
{
  Powtarzaj az do chwili gdy trafisz na element istniejacy
  w Z1 i nie bedzie go w Z2 lub miniesz koniec
}
   I:=0; Podzbior:=false;
   repeat I:=I+1 until (Z1[I] and not Z2[I]) or (I>Moc);
   if (I>Moc) then Podzbior:=true
end;



procedure WypiszZbior(var Z: Zbior);
(* Wypisuje niezerowe elementy zbioru Z
          w postaci liczb calkowitych *)
var
  I, Licz: integer;
begin
  Licz:=0;
  writeln;
  for I:=1 to Moc do
  begin
     if Z[I] then
     begin
       write(I:5,', ');
       Licz:=Licz+1;
       if(Licz mod 10) =0 then writeln
     end
  end;
  writeln
end;

{---------------------------------------}

var
  S, Q: Zbior;
  E: ElementZbioru;

begin
  clrscr;
  UtworzPusty(S);
  for E:=7 to 46 do
     DodajElement(E,S);
  WypiszZbior(S);
  writeln(PustyZbior(S));
{
  Ci/agle musimy budowa/c zbi/or, a wlasciwie inicjowa/c go
  uzywajac procedury UtworzPusty.
  Jest to do/s/c uci/a/zliwe. Czy mo/zna to w jaki/s
  sposob upro/sci/c? Pomagaja tutaj obiekty!
}
  ZerujZbior(Q);
```

```
  DodajElement(30,Q);
      writeln(RowneZbiory(S,Q));
  DodajElement(40,Q);
      writeln(RowneZbiory(S,Q));
  RoznicaZbiorow(S,Q,S);
  WypiszZbior(S);
  writeln(Podzbior(Q,S));
  SumaZbiorow(S,Q,S);
  WypiszZbior(S);
  writeln(Podzbior(Q,S))
end.
```