podstawy i języki programowania

pjp

Podstawy programowania

Część siódma

Tablice — różne zastosowania

Zbiory

Autor

Roman Simiński

Kontakt

siminski@us.edu.pl
www.us.edu.pl/~siminski

Niniejsze opracowanie zawiera skrót treści wykładu, lektura tych materiałów nie zastąpi uważnego w nim uczestnictwa.
Opracowanie to jest chronione prawem autorskim. Wykorzystywanie jakiegokolwiek fragmentu w celach innych niż nauka własna jest nielegalne.
Dystrybuowanie tego opracowania lub jakiejkolwiek jego części oraz wykorzystywanie zarobkowe bez zgody autora jest zabronione.

Problem 0

Do programu „dopływa” strumień znaków — np. użytkownik wpisuje z klawiatury znak po znaku.

Zadaniem programu jest wyznaczenie, ile wśród tych znaków *jest dużych liter*.

Zakładamy, że wpisanie przez użytkownika znaku '\' kończy program.

```
Podawaj kolejne znaki (kazdy zatwierdz Enterem),
a ja policze ile wpisales duzych liter.
Nacisnij \ by zakonczyc.
>a
>b
>c
>A
>B
>C
>\
Liczba duzych liter: 3
Nacisnij Enter by zakonczyc program...
```

Copyright © Roman Simiński

Problem 0 — analiza + pseudokod

```
Zdefiniuj zmienną całkowitą  — licznik dużych liter: LiczDL

Wyzeruj LiczDL

Powtarzaj
  Wczytaj znak
  Jeżeli znak jest dużą literą to
    Zwiększ LiczDL o 1
Aż zostanie naciśnięty znak '\'

Wyprowadź LiczDL
```

Copyright © Roman Simiński

Problem 0 — przykładowe rozwiązanie

```
Program LiczbaDuzychLiter;
Var
  LiczDL : Integer; (* Licznik duzych liter *)
  Znak   : Char;    (* Przechowuje wczytany znak *)
Begin
  WriteLn( 'Podawaj kolejne znaki (kazdy zatwierdz Enterem), ' );
  WriteLn( 'a ja policze ile wpisales duzych liter.' );
  WriteLn( 'Nacisnij \ by zakonczyc.' );

  (* Zerowanie licznika - bardzo wazna operacja *)
  LiczDL := 0;

  (* Iteracja wczytujaca znaki i zliczajaca duze litery *)
  Repeat
    Write( '>');
    ReadLn( Znak );
    If Znak In [ 'A' .. 'Z' ] Then
      LiczDL := LiczDL + 1;
  Until Znak = '\';

  (* Wyswietlenie licznosci duzych liter *)
  WriteLn( 'Liczba duzych liter: ', LiczDL );
  Write( 'Nacisnij Enter by zakonczyc program...' );
  ReadLn;
End.
```

Copyright © Roman Simiński

Problem 0 — alternatywne wersje testu na dużą literę

Wersja z wykorzystaniem zbioru znaków — testowanie przynależności znaku zapamiętanego w zmiennej *Znak* do zbioru dużych liter ['A' .. 'Z']

```
If Znak In [ 'A' .. 'Z' ] Then
  LiczDL := LiczDL + 1;
```

Testowanie, czy znak zapamiętany w zmiennej *Znak*, zawiera się w przedziale dużych liter typu porządkowego *Char*.

```
If ( Znak >= 'A' ) And ( Znak <= 'Z' ] Then
  LiczDL := LiczDL + 1;
```

Copyright © Roman Simiński

Problem 1

Jak w Problem 0, tylko wyznaczamy liczbę wystąpień każdej z dużych liter.

```
Podawaj kolejne znaki (kazdy zatwierdz Enterem),
a ja policze ile wpisales duzych liter.
Nacisnij \ by zakonczyc.
>A
>B
>C
>A
>B
>\

Duze litery:
¦A:    2¦B:    2¦C:    1¦D:    0¦E:    0¦F:    0¦G:    0¦H:    0¦I:    0¦J:    0
¦K:    0¦L:    0¦M:    0¦N:    0¦O:    0¦P:    0¦Q:    0¦R:    0¦S:    0¦T:    0
¦U:    0¦V:    0¦W:    0¦X:    0¦Y:    0¦Z:    0
Nacisnij Enter by zakonczyc program
```

Copyright © Roman Simiński

Problem 1, rozwiązanie naiwne — analiza + pseudokod

```
Zdefiniuj zmienną całkowitą — licznik dla litery A: LiczA
Zdefiniuj zmienną całkowitą — licznik dla litery B: LiczB
...
Zdefiniuj zmienną całkowitą — licznik dla litery Z: LiczZ

Wyzeruj LiczA, LiczB, ... LiczZ

Powtarzaj
  Wczytaj znak
  Jeżeli znak jest dużą literą to
     Jeżeli znak to 'A' to
        Zwiększ LiczA o 1
     Jeżeli znak to 'B' to
        Zwiększ LiczB o 1
     ...
     Jeżeli znak to 'Z' to
        Zwiększ LiczZ o 1
Aż zostanie naciśnięty znak '\'

Wyprowadź LiczA, LiczB, ... LiczZ
```

Copyright © Roman Simiński

Problem 1, rozwiązanie naiwne — wątpliwości

Zdefiniuj zmienną całkowitą — licznik dla litery A: LiczA
Zdefiniuj zmienną całkowitą — licznik dla litery B: LiczB
...
Zdefiniuj zmienną całkowitą — licznik dla litery Z: LiczZ

Wyzeruj LiczA, LiczB, ... LiczZ

Potarzaj
  Wczytaj znak
  Jeżeli znak jest dużą literą to
    Jeżeli znak to 'A' to
      Zwiększ LiczA o 1
    Jeżeli znak to 'B' to
      Zwiększ LiczB o 1
    ...
    Jeżeli znak to 'Z' to
      Zwiększ LiczZ o 1
Aż zostanie naciśnięty znak '\'

Wyprowadź LiczA, LiczB, ... LiczZ

Copyright © Roman Simiński

Problem 1, rozwiązanie naiwne — wątpliwości, cd. ...

Zdefiniuj zmienną całkowitą — licznik dla litery A: LiczA
Zdefiniuj zmienną całkowitą — licznik dla litery B: LiczB
...
Zdefiniuj zmienną całkowitą — licznik dla litery Z: LiczZ

Wyzeruj LiczA, LiczB, ... LiczZ

Potarzaj
  Wczytaj znak
  Jeżeli znak jest dużą literą to
    Jeżeli znak to 'A' to
      Zwiększ LiczA o 1
    Jeżeli znak to 'B' to
      Zwiększ LiczB o 1
    ...
    Jeżeli znak to 'Z' to
      Zwiększ LiczZ o 1
Aż zostanie naciśnięty znak '\'

Wyprowadź LiczA, LiczB, ... LiczZ

Copyright © Roman Simiński

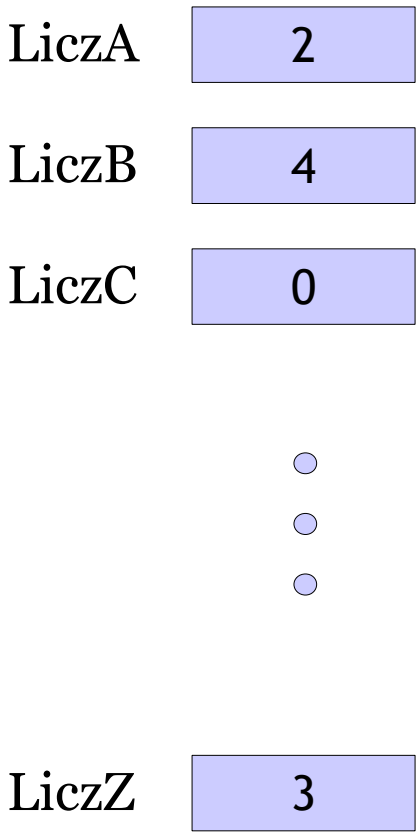
Elementarz programisty: równanie Niklausa Wirth'a

algorytmy + struktury danych = programy

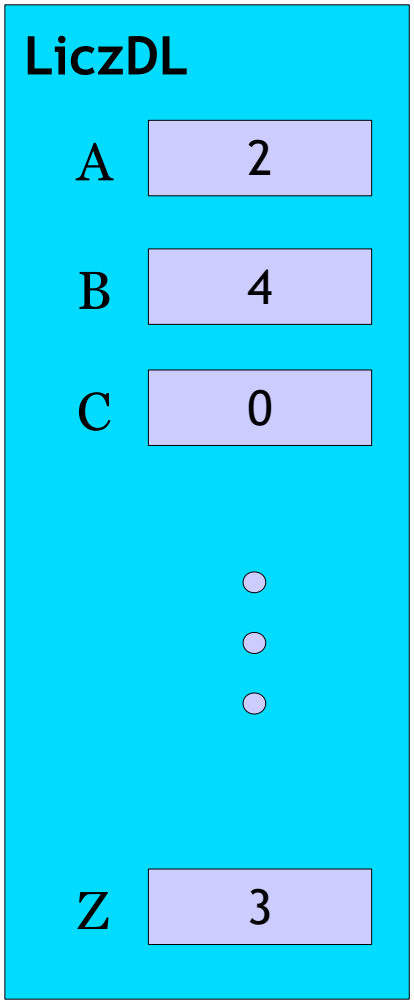
Copyright © Roman Simiński

Problem 1, poszukiwanie rozwiązanie nienaiwnego

Rozwiązanie naiwne: definiujemy tyle liczników, ile dużych liter, każdy licznik jest osobną zmienną.



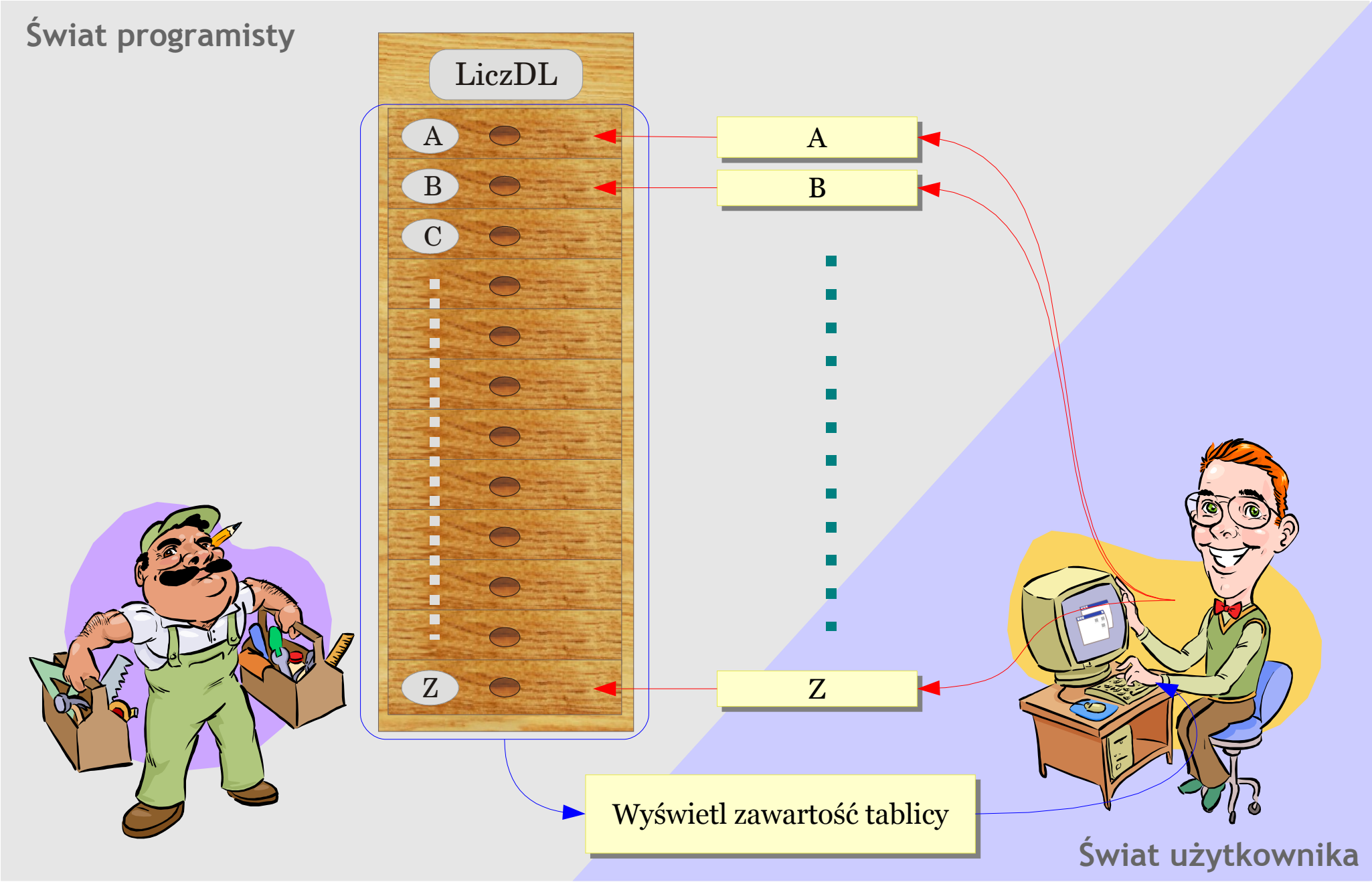
Rozwiązanie lepsze: definiujemy tablicę liczb całkowitych, tablica ma tyle elementów ile dużych liter.



Każdy element tablicy będzie licznikiem dla danej litery

Copyright © Roman Simiński

Problem 1, poszukiwanie rozwiązanie nienaiwnego, cd. ...



Copyright © Roman Simiński

Problem 1, jak zadeklarować tablicę liczników dla dużych liter?

Deklaracja zmiennej tablicowej *LiczDL*:

```
Var
  LiczDL : Array[ 'A' .. 'Z' ] Of Integer;
```

Bardziej elegancka forma:

```
Type
  DuzeLitery = 'A' .. 'Z';
Var
  LiczDL : Array[ DuzeLitery ] Of Integer;
```

Jak odwołujemy się do elementów tablicy?

```
LiczDL[ 'A' ] := 0;
...
LiczDL[ 'A' ] := LiczDL[ 'A' ] + 1;
...
WriteLn( LiczDL[ 'A' ] );
```

Copyright © Roman Simiński

Problem 1 — rozwiązanie sprytne

Rozwiązanie bazuje na tym samy algorytmie co rozwiązanie naiwne, „sprytność” rozwiązania polega na mądrym wykorzystaniu możliwości typów tablicowych.

Zakładając, ze zmienna *Znak* jest typu *Char*, zerowanie wszystkich liczników:

```
(* Zerowanie tablicy licznikow duzych liter *)
For Znak := 'A' To 'Z' Do
  LiczDL[ Znak ] := 0;
```

Co rozwija się w trakcie wykonania w ciąg instrukcji:

```
LiczDL[ 'A' ] := 0;
LiczDL[ 'B' ] := 0;
LiczDL[ 'C' ] := 0;
.
.
.
LiczDL[ 'Z' ] := 0;
```

Copyright © Roman Simiński

Problem 1 — rozwiązanie sprytne

Wczytanie znaku do zmiennej *Znak*, sprawdzenie czy to duża litera, oraz zwiększenie licznika odpowiadającego wczytanej literze.

```
Write( '>');
ReadLn( Znak );
If Znak In [ 'A' .. 'Z' ] Then
  LiczDL[ Znak ] := LiczDL[ Znak ] + 1;
```

To samo powtarzane, aż do wczytania znacznika końca:

```
Repeat

  Write( '>');
  ReadLn( Znak );
  If Znak In [ 'A' .. 'Z' ] Then
    LiczDL[ Znak ] := LiczDL[ Znak ] + 1;

Until Znak = '\';
```

```
Podawaj kolejne znaki (kazdy zatwierdz Enterem),
a ja policze ile wpisales duzych liter.
Nacisnij \ by zakonczyc.
>A
>B
>C
>A
>B
>\
```

Copyright © Roman Simiński

Problem 1 — rozwiązanie sprytne

Wyprowadzenie zawartości tablicy liczników dużych liter do strumienia wyjściowego programu:

```
WriteLn( 'Duze litery:' );

For Znak := 'A' To 'Z' Do
  Write( '|', Znak , ': ', LiczDL[ Znak ] : 4 );
```

```
Podawaj kolejne znaki (kazdy zatwierdz Enterem),
a ja policze ile wpisales duzych liter.
Nacisnij \ by zakonczyc.
>A
>B
>C
>A
>B
>\

Duze litery:
|A:    2|B:    2|C:    1|D:    0|E:    0|F:    0|G:    0|H:    0|I:    0|J:    0
|K:    0|L:    0|M:    0|N:    0|O:    0|P:    0|Q:    0|R:    0|S:    0|T:    0
|U:    0|V:    0|W:    0|X:    0|Y:    0|Z:    0
Nacisnij Enter by zakonczyc program
```

Copyright © Roman Simiński

Problem 1 — przykładowy kod programu

```
Program LiczenieDuzychLiter;
Type
  DuzeLitery = 'A' .. 'Z';
Var
  LiczDL : Array[ DuzeLitery ] Of Integer;
  Znak   : Char;
Begin
  (* Informacje o programie *)
  WriteLn( 'Podawaj kolejne znaki (kazdy zatwierdz Enterem), ' );
  WriteLn( 'a ja policze ile wpisales duzych liter.' );
  WriteLn( 'Nacisnij \ by zakonczyc.' );

  (* Zerowanie tablicy licznikow duzych liter *)
  For Znak := 'A' To 'Z' Do
    LiczDL[ Znak ] := 0;

  (* Iteracja wczytujaca i zliczajaca *)
  Repeat
    Write( '>');
    ReadLn( Znak );
    If Znak In [ 'A' .. 'Z' ] Then
      LiczDL[ Znak ] := LiczDL[ Znak ] + 1;
  Until Znak = '\';

                                                  (* Ciag dalszy >> *)
```

Copyright © Roman Simiński

Problem 1 — rozwiązanie sprytne, przykładowy kod programu

```
  (* Wyswietlenie liczby kazdej z duzych liter *)
  WriteLn;
  WriteLn( 'Duze litery:' );
  For Znak := 'A' To 'Z' Do
    Write( '|', Znak , ': ', LiczDL[ Znak ] : 4 );

  WriteLn;
  Write( 'Nacisnij Enter by zakonczyc program' );
  ReadLn;
End.
```

Dobór odpowiedniej struktury danych pozwolił zastosować naiwny i — zdawało by się — nieoptymalny algorytm.

Często jest tak, że dobry pomysł na program nie tylko tkwi w algorytmie ale w dobrze zorganizowanych strukturach danych.

Copyright © Roman Simiński

Problem 2

Jak w Problem 1, tylko wyznaczamy liczbę wystąpień każdej z liter dużych, małych i cyfr.

```
Podawaj kolejne znaki (kazdy zatwierdz Enterem), a ja policze
ile wpisales kazdej z liter oraz kazdej cyfr.
Nacisnij \ by zakonczyc.
>a
>a
>b
>A
>A
>B
>0
>1
>\

Duze litery:
!A:    2!B:    1!C:    0!D:    0!E:    0!F:    0!G:    0!H:    0!I:    0!J:    0
!K:    0!L:    0!M:    0!N:    0!O:    0!P:    0!Q:    0!R:    0!S:    0!T:    0
!U:    0!V:    0!W:    0!X:    0!Y:    0!Z:    0
Male litery:
!a:    2!b:    1!c:    0!d:    0!e:    0!f:    0!g:    0!h:    0!i:    0!j:    0
!k:    0!l:    0!m:    0!n:    0!o:    0!p:    0!q:    0!r:    0!s:    0!t:    0
!u:    0!v:    0!w:    0!x:    0!y:    0!z:    0
Cyfry:
!0:    1!1:    1!2:    0!3:    0!4:    0!5:    0!6:    0!7:    0!8:    0!9:    0

Nacisnij Enter by zakonczyc program
```

Copyright © Roman Simiński

Problem 2 — analogicznie jak dla dużych liter?

Deklaracja zmiennej tablicowej *LiczDL, LiczML, LiczC*:

```
Var
  LiczDL : Array[ 'A' .. 'Z' ] Of Integer;
  LiczML : Array[ 'a' .. 'z' ] Of Integer;
  LiczC  : Array[ '0' .. '9' ] Of Integer;
```

Bardziej elegancka forma:

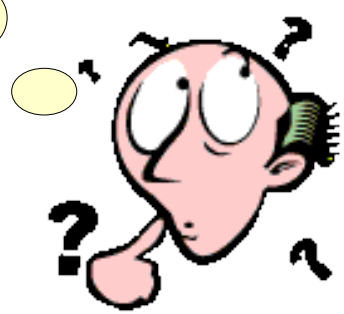
```
Type
  DuzeLitery = 'A' .. 'Z';
  MaleLitery = 'a' .. 'z';
  Cyfry      = '0' .. '9';

Var
  LiczDL : Array[ DuzeLitery ] Of Integer;
  LiczML : Array[ MaleLitery ] Of Integer;
  LiczC  : Array[   Cyfry    ] Of Integer;
```

Copyright © Roman Simiński

Problem 2 — analogicznie jak dla dużych liter? Wątpliwości...

Niby sprytne, ale będzie teraz trzeba męczyć się z trzema tablicami... :(

A jeżeli jeszcze dojdzie zapamiętywanie liczebności innych znaków, to robi się nieciekawie!



Copyright © Roman Simiński

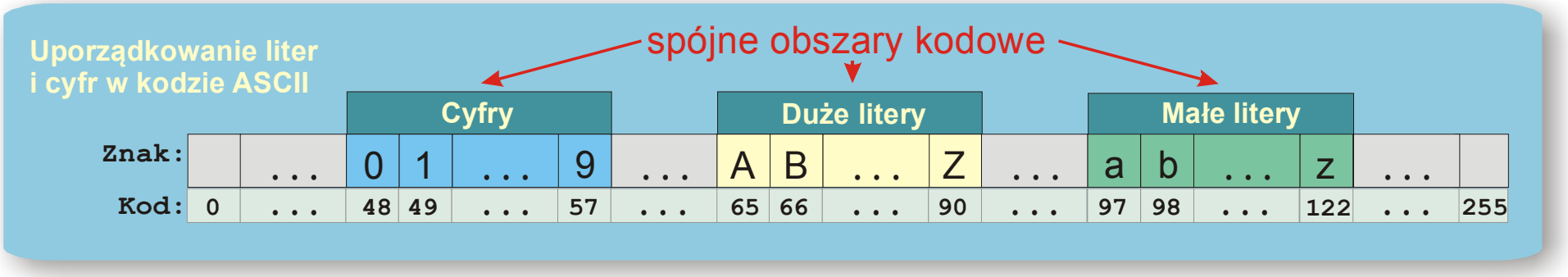
Problem 2 — a może wystarczy jedna tablica?

Deklaracja tablicy liczników dla wszystkich znaków typy *Char*:

```
Var
  Licznik : Array[ Char ] Of Integer;
```

Przy tak zdefiniowanej tablicy, każdy znak będący elementem typu *Char* posiada swój własny licznik. Można zatem zliczać ile jest liter, cyfr, przecinków, kropek, itp., itd.

Dla naszych potrzeb wystarczy wykorzystać tylko wybrane elementy tablicy, reszta póki co się marnuje. Ale mimo tej redundancji, przetwarzanie jest proste, dotyczy jednej tablicy.



Copyright © Roman Simiński

Problem 2 — wystarczy jedna tablica

```
Program LiczenieLiterICyfr;
Var
  Licznik : Array[ Char ] Of Integer;
  Znak    : Char;
Begin
  (* Informacje o programie *)
  WriteLn( 'Podawaj kolejne znaki (kazdy zatwierdz Enterem), a ja policze' );
  WriteLn( 'ile wpisales kazdej z liter oraz kazdej cyfr.' );
  WriteLn( 'Nacisnij \ by zakonczyc.' );

  (* Zerowanie tablicy licznikow duzych liter, dziala dla ASCII *)
  For Znak := Chr( 0 ) To Chr( 255 ) Do
    Licznik[ Znak ] := 0;

  (* Iteracja wczytujaca i zliczajaca *)
  Repeat
    Write( '>');
    ReadLn( Znak );
    Licznik[ Znak ] := Licznik[ Znak ] + 1;
  Until Znak = '\';
```

(* Ciag dalszy >> *)

Copyright © Roman Simiński

Problem 2 — wystarczy jedna tablica

```
  (* Wyswietlenie licznika dla kazdej z duzych liter *)
  WriteLn;
  WriteLn( 'Duze litery:' );
  For Znak := 'A' To 'Z' Do
    Write( '|', Znak , ': ', Licznik[ Znak ] : 4 );

  (* Wyswietlenie licznika dla kazdej z malych liter *)
  WriteLn;
  WriteLn( 'Male litery:' );
  For Znak := 'a' To 'z' Do
    Write( '|', Znak , ': ', Licznik[ Znak ] : 4 );

  (* Wyswietlenie licznika dla kazdej z cyfr *)
  WriteLn;
  WriteLn( 'Cyfry:' );
  For Znak := '0' To '9' Do
    Write( '|', Znak , ': ', Licznik[ Znak ] : 4 );

  WriteLn;
  Write( 'Nacisnij Enter by zakonczyc program' );
  ReadLn;
End.
```

Copyright © Roman Simiński

Kolejny program — wytypuj szóstkę do „dużego lotka”

Zadaniem programu jest wylosowanie sześciu liczb, pozwalających wypełnić pojedynczy typowanie tzw. „dużego lotka”.

Losujemy liczby z przedziału 1 .. 49, w wylosowanej szóstce nie może być powtórzeń.

```
Losuje szostke liczb z 49-ciu.
Wylosowana szostka:
4 9 17 31 37 43
Nacisnij Enter by zakonczyc program...
```

Copyright © Roman Simiński

Wytypuj szóstkę do „dużego lotka” — analiza + pseudokod

```
Zdefiniuj zmienną całkowitą  — licznik losowanych liczb: Wylosowanych
Zdefiniuj zmienną całkowitą  — pamięta aktualnie wylosowaną liczbę: Liczba

Zainicjuj generator liczb pseudolosowych
Wyzeruj zmienna Wylosowanych

Powtarzaj
   Wylosuj wartość z przedzialu 1 .. 49 i wstaw do zmiennej Liczba
   Wyprowadź zawartość zmiennej Liczba do strumienia wyjściowego
   Zwiększ zmienną Wylosowanych o 1
Aż zmienna Wylosowanych będzie miała wartość 6
```

To ma szansę zadziałać, ale nigdy nie wiadomo, czy losując kolejną liczbę, nie otrzymamy czasem którejś z już wcześniej wylosowanych liczb..., ale umówmy się, że ten problem rozwiążemy później.

Copyright © Roman Simiński

Wytypuj szóstkę do „dużego lotka” — koślawa, pierwsza wersja

```
Program LosujSzostke;
Const
  (* Losujemy liczby od 1 do wartosci stalej Zakres *)
  Zakres = 49;
Var
  Liczba        : Integer; (* Losowana liczba, kandydat do szostki *)
  Wylosowanych  : Integer; (* Licznik losowanych liczb  *)

Begin
  WriteLn( 'Losuje szostke liczb z 49-ciu.' );
  Randomize; (* Inicjalizacja generatora liczb pseudolosowych *)

  WriteLn( 'Wylosowana szostka: ' );
  Wylosowanych := 0; (* Zerowanie licznika wylosowanych liczb *)
  Repeat
    Liczba := Random( Zakres ) + 1;  (* Losowanie liczby z zakresu *)
    Write( Liczba, ' ' ); (* Wyprowadzenie wylosowanej liczby *)
    Wylosowanych := Wylosowanych + 1;
  Until Wylosowanych = 6; (* Powtarzamy az wylosujemy szostke *)

  WriteLn;
  WriteLn( 'Nacisnij Enter by zakonczyc program...' );
  ReadLn;
End.
```

```
Wylosowana szostka:
35 42 9 9 20 32
Nacisnij Enter by zakonczyc program...
```

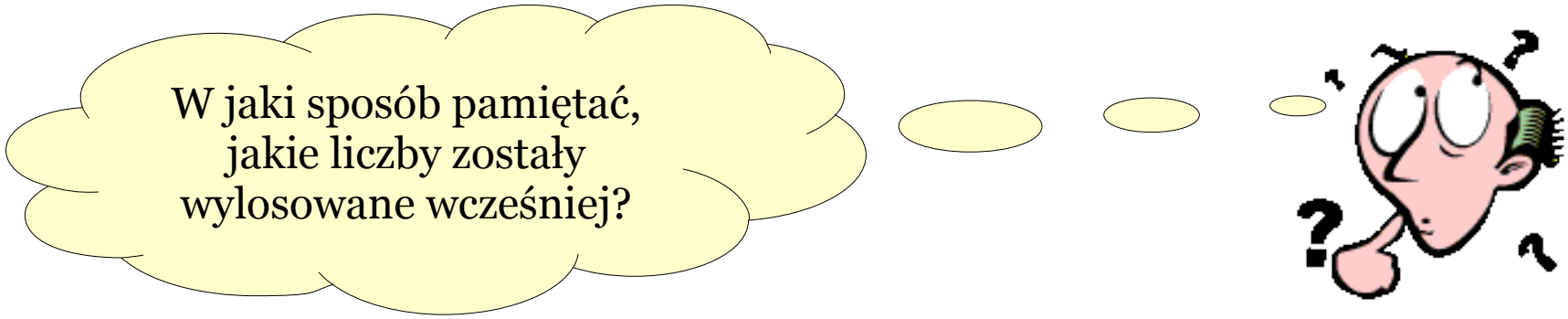
Copyright © Roman Simiński

Wytypuj szóstkę do „dużego lotka” — jak wyeliminować duplikaty?

Zdefiniuj zmienną całkowitą — licznik losowanych liczb: Wylosowanych
Zdefiniuj zmienną całkowitą — pamięta aktualnie wylos. liczbę: Liczba

Zainicjuj generator liczb pseudolosowych
Wyzeruj zmienna Wylosowanych

Potarzaj
  Wylosuj wartość z przedzialu 1 .. 49 i wstaw do zmiennej Liczba
  Jeżeli wartość zmiennej Liczba nie była jeszcze wylosowana To
    Wyprowadź zawartość zmiennej Liczba
    Zwiększ zmienną Wylosowanych o 1
Aż zmienna Wylosowanych będzie miała wartość 6

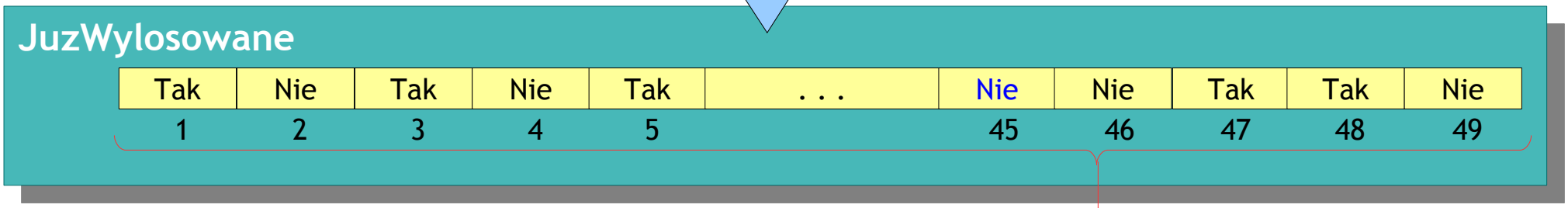


Copyright © Roman Simiński

Wytypuj szóstkę do „dużego lotka” — rejestr wylosowanych liczb

Każdy element rejestru odpowiada liczbie — kandydatce do szóstki. Jeżeli na danej pozycji w rejestrze ustawiona jest wartość *Tak*, to dana liczba została już wylosowana.

Jeżeli na zadanej pozycji jest wartość *Nie*, taka liczba jeszcze nie była wylosowana.



Załóżmy, że wylosowano liczbę 45.

Na pozycji tej liczbie odpowiadającej jest *Nie*, zatem takiej liczby jeszcze nie wylosowano.

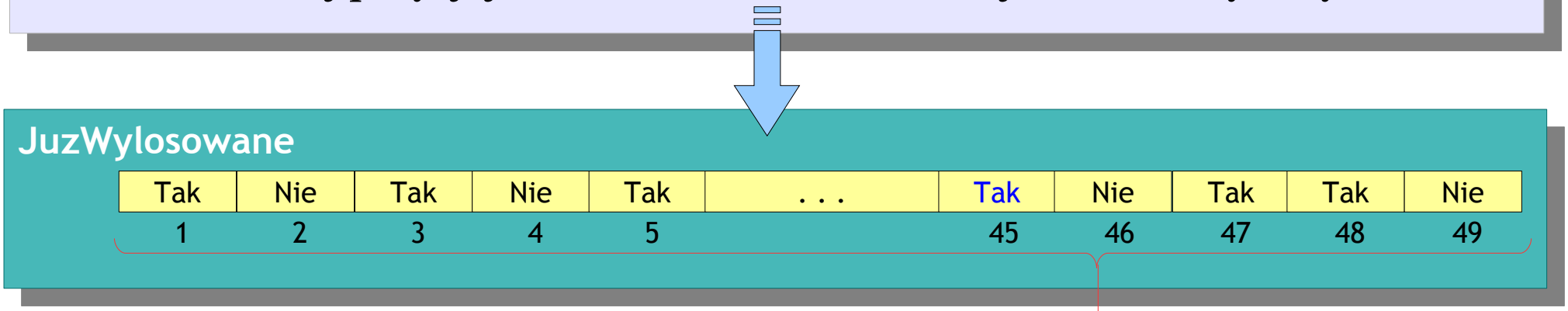
Liczba 45 może wejść do losowanej szóstki.

Copyright © Roman Simiński

Wytypuj szóstkę do „dużego lotka” — rejestr wylosowanych liczb

Każdy element rejestru odpowiada liczbie — kandydatce do szóstki. Jeżeli na danej pozycji w rejestrze ustawiona jest wartość *Tak*, to dana liczba została już wylosowana.

Jeżeli na zadanej pozycji jest wartość *Nie*, taka liczba jeszcze nie była wylosowana.



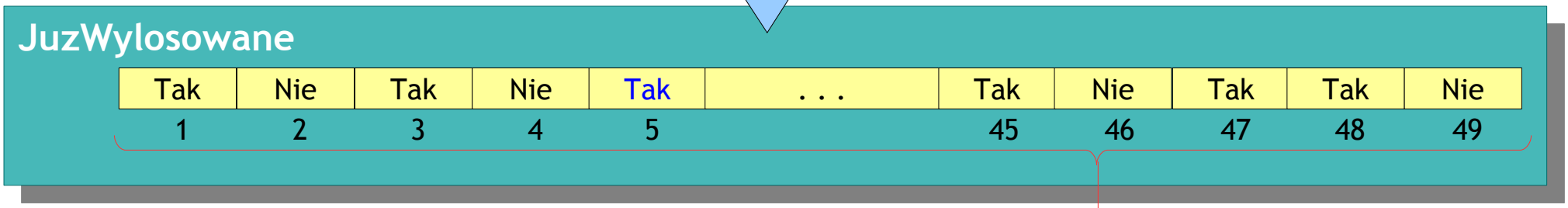
Zaznaczamy, że liczba 45 została wylosowana wpisując do rejestru *Tak*, na pozycji odpowiadającej tej liczbie.

Copyright © Roman Simiński

Wytypuj szóstkę do „dużego lotka” — rejestr wylosowanych liczb

Każdy element rejestru odpowiada liczbie — kandydatce do szóstki. Jeżeli na danej pozycji w rejestrze ustawiona jest wartość *Tak*, to dana liczba została już wylosowana.

Jeżeli na zadanej pozycji jest wartość *Nie*, taka liczba jeszcze nie była wylosowana.



Załóżmy, że wylosowano liczbę 5.

Na pozycji tej liczbie odpowiadającej jest *Tak*, zatem tę liczbę już wylosowano.

Liczba 5 zostaje odrzucona jako dublet.

Copyright © Roman Simiński

Wytypuj szóstkę do „dużego lotka” — rejestr wylosowanych liczb

Każdy element rejestru odpowiada liczbie — kandydatce do szóstki. Jeżeli na danej pozycji w rejestrze ustawiona jest wartość *Tak*, to dana liczba została już wylosowana.

Jeżeli na zadanej pozycji jest wartość *Nie*, taka liczba jeszcze nie była wylosowana.

JuzWylosowane

Tak	Nie	Tak	Nie	Tak	. . .	Tak	Nie	Tak	Tak	Nie
1	2	3	4	5		45	46	47	48	49

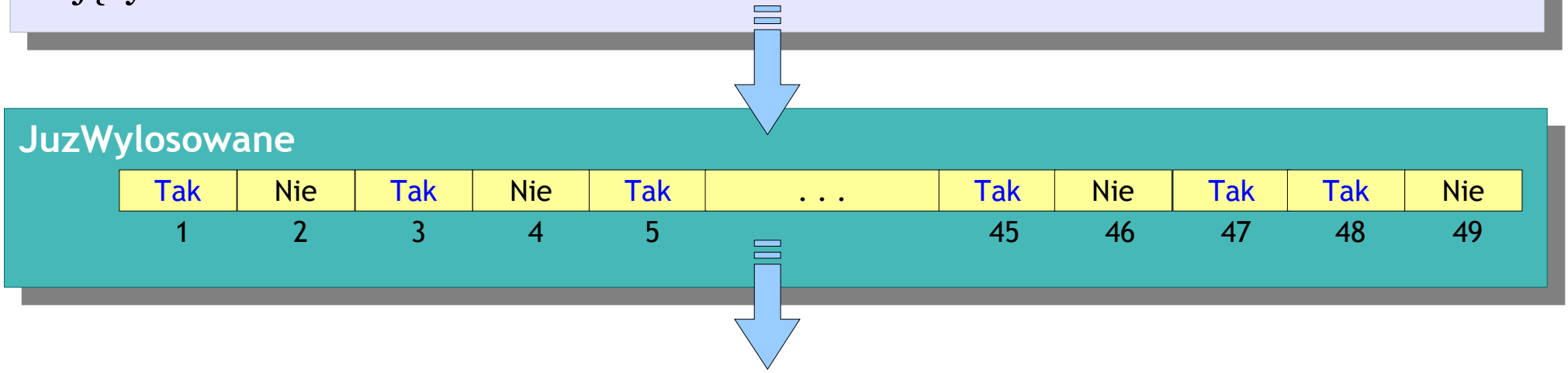
Z rejestru wylosowanych liczb można „wyczytać”, które z nich zostały wylosowane:

1 3 5 45 47 48

Copyright © Roman Simiński

Jak uzyskać taki rejestr w programie?

Taki rejestr to tablica 49 elementów. Każdy element takiej tablicy przyjmuje tylko jedną z dwóch możliwych wartości {*Tak*, *Nie*}, odpowiada to typowi *Boolean* w języku Pascal.




```
Const
  Zakres = 49;

Var
  JuzWylosowane : Array[ 1 .. Zakres ] Of Boolean;
```

Copyright © Roman Simiński

Jak sprawdzić, czy liczba już była (lub nie była) wylosowana?

```
(* Losowanie liczby z zakresu *)
Liczba := Random( Zakres ) + 1;

(* Czy juz te liczbe wylosowano? *)
If JuzWylosowane[ Liczba ] Then
  Begin
    (* Cos tam, gdy byla wylosowana *)
  End;
```

```
(* Losowanie liczby z zakresu *)
Liczba := Random( Zakres ) + 1;

(* Czy jeszcze tej liczby nie wylosowano? *)
If Not JuzWylosowane[ Liczba ] Then
  Begin
    (* Cos tam, gdy nie byla wylosowana *)
  End;
```

Copyright © Roman Simiński

Jak oznaczyć, że liczba zostala wylosowana?

```
(* Losowanie liczby z zakresu *)
Liczba := Random( Zakres ) + 1;

(* Czy jeszcze tej liczby nie wylosowano? *)
If Not JuzWylosowane[ Liczba ] Then
  Begin
    (* Nie bylo jeszcze takiej liczby, ustawiamy jej "flage" w tablicy *)
    JuzWylosowane[ Liczba ] := True;

    (* Cos tam dalej, gdy nie byla wylosowana *)
  End;
```

Copyright © Roman Simiński

Jak wylosować całą szóstkę?

```
(* Zerowanie licznika wylosowanych liczb *)
Wylosowanych := 0;

Repeat

  (* Losowanie liczby z zakresu *)
  Liczba := Random( Zakres ) + 1;

  (* Czy czasem juz tej liczby nie wylosowano ? *)
  If Not JuzWylosowane[ Liczba ] Then
    Begin
      (* Nie bylo jeszcze tej liczby, ustawiamy jej "flage" w tablicy *)
      JuzWylosowane[ Liczba ] := True;

      (* Zwiekszamy licznik wylosowanych liczb *)
      Wylosowanych := Wylosowanych + 1;
    End;

    (* Powtarzamy az wylosujemy szostke *)
Until Wylosowanych = 6;
```

Copyright © Roman Simiński

Jak wyprowadzić szóstkę?

```
(* Wyswietlenie wylosowanej szostki *)
WriteLn( 'Wylosowana szostka: ' );

For Liczba := 1 To Zakres Do  (* Przejrzyj rejestr liczb z zakresu *)
    If JuzWylosowane[ Liczba ] Then (* Jezeli liczba wylosowana *)
      Write( Liczba, ' ' ); (* Wyprowadz liczbe *)
```

Uwaga, żeby to wszystko zadziałało, trzeba wcześniej wyzerować rejestr wylosowanych liczb:

```
(* Jeszcze nie wylosowano zadnej liczby, czyszczenie tablicy *)
For Liczba := 1 To Zakres Do
  JuzWylosowane[ Liczba ] := False;
```

Copyright © Roman Simiński

Wytypuj szóstkę do „dużego lotka” – pełna wersja programu

```
Program LosujSzostke;
Const
  (* Losujemy liczby od jeden do wartosci stalej Zakres *)
  Zakres = 49;
Var
  (* Tablica "flag" dla kazdej liczby z zakresu. Jezeli liczba zostala    *)
  (* wylosowana, to na jej pozycji bedzie wartosc True, False gdy liczba  *)
  (* nie zostala wylosowana                                               *)
  JuzWylosowane : Array[ 1 .. Zakres ] Of Boolean;

  (* Tutaj wylosowana liczba, kandydat do losowanej szostki *)
  Liczba        : Integer;

  (* Licznik losowanych liczb  *)
  Wylosowanych  : Integer;

Begin
  WriteLn( 'Losuje szostke liczb z 49-ciu.' );

  (* Jeszcze nie wylosowano zadnej liczby, czyszczenie tablicy *)
  For Liczba := 1 To Zakres Do
    JuzWylosowane[ Liczba ] := False;

  (* Inicjalizacja generatora liczb pseudolosowych *)
  Randomize;                                                (* Ciag dalszy >> *)
```

Copyright © Roman Simiński

Wytypuj szóstkę do „dużego lotka” — pełna wersja programu, cd.

```
  (* Zerowanie licznika wylosowanych liczb *)
  Wylosowanych := 0;
  Repeat
    Liczba := Random( Zakres ) + 1; (* Losowanie liczby z zakresu *)
    If Not JuzWylosowane[ Liczba ] Then (* Czy wczesniej nie wylosowana? *)
      Begin
        (* Jeszcze niewylosowana, ustawiamy jej "flage" w tablicy *)
        JuzWylosowane[ Liczba ] := True;
        (* Zwiekszamy licznik wylosowanych liczb *)
        Wylosowanych := Wylosowanych + 1;
      End;
      (* Powtarzamy az wylosujemy szostke *)
  Until Wylosowanych = 6;

  (* Wyswietlenie wylosowanej szostki *)
  WriteLn( 'Wylosowana szostka: ' );
  For Liczba := 1 To Zakres Do  (* Sprawdz flagi liczb z zakresu *)
    If JuzWylosowane[ Liczba ] Then (* Jezeli liczba zostala wylosowana *)
      Write( Liczba, ' ' ); (* Wyprowadz liczbe *)

  WriteLn;
  WriteLn( 'Nacisnij Enter by zakonczyc program...' );
  ReadLn;
End.
```

Copyright © Roman Simiński

Wytypuj szóstkę do „dużego lotka” — wersja alternatywna

Zdefiniuj zmienną całkowitą — licznik losowanych liczb: Wylosowanych
Zdefiniuj zmienną całkowitą — pamięta aktualnie wylos. liczbę: Liczba
Zdefiniuj zmienną będącą zbiorem liczb — pamięta wylosowane liczby: Wylosowane

Zainicjuj generator liczb pseudolosowych
Wylosowane := $\emptyset$

Potarzaj
  Wylosuj wartość z przedzialu 1 .. 49 i wstaw do zmiennej Liczba
  Jeżeli Liczba $\notin$ Wylosowane To
    Wylosowane := Wylosowane $\vee$ Liczba
    Zwiększ zmienną Wylosowanych o 1
Aż zmienna Wylosowanych = 6

Dla każdej liczby z zakresu Wykonaj
  Jeżeli Liczba $\in$ Wylosowane To
    Wyprowadź Liczba

Copyright © Roman Simiński

Zbiór jak struktura danych

- Typ zbiorowy należy do złożonych (strukturalnych) typów danych.
- Zbiór (zmienna teoriomnogościowa) składa się z *elementów tego samego typu podstawowego*, musi to być *typ porządkowy*.
- Zbiór może być pusty, lub zawierać elementy.
- Liczba elementów zbioru *może się zmieniać*.
- Komputerowa implementacja zbiorów *ma ograniczenia*. Liczność zbioru nie może przekraczać pewnego limitu, zwykle jest to 256 elementów.

Zdefiniowanie takiej zmiennej się nie uda. Typ *Integer* ma zbyt wiele wartości:

```
Var
  Wylosowane : Set of Integer;
```

Zdefiniowanie takiej zmiennej się powiedzie. Typ *Char* ma mniej wartości:

```
Var
  DozwoloneZnaki : Set of Char;
```

Copyright © Roman Simiński

Definiowanie zbiorów

Typ podstawowy musi być typem porządkowym.

```
Var
  TrzyLitery : Set Of 'A' .. 'C';
```

Zmienna TrzyLitery może mieć wartości:

```
[], [ 'A' ], [ 'B' ], [ 'C' ], [ 'A', 'C' ], [ 'A', 'B' ], [ 'B', 'C' ],
[ 'A', 'B, 'C' ]
```

Wszystkie możliwe podzbiory zbioru podstawowego to *zbiór potęgowy*,

Copyright © Roman Simiński

Operacje na zbiorach

Ustawienie zbioru pustego:

```
TrzyLitery := [];
```

Dopisanie elementu do zbioru:

```
TrzyLitery := TrzyLitery + [ 'A' ];
```

Sprawdzenie, czy elemet należy do zbioru:

```
Var
  Litera : 'A' .. 'C';
.
.
.
If Liter In TrzyLitery Then
  { cos tam };
```

Na zbiorach działają operatory + (suma zbiorów), — (różnica zbiorów), * (przecięcie zbiorów), **In** (przynależność elementu do zbioru), = (równość zbiorów), <> (nierówność zbiorów), >= oraz <= (zawieranie zbiorów).

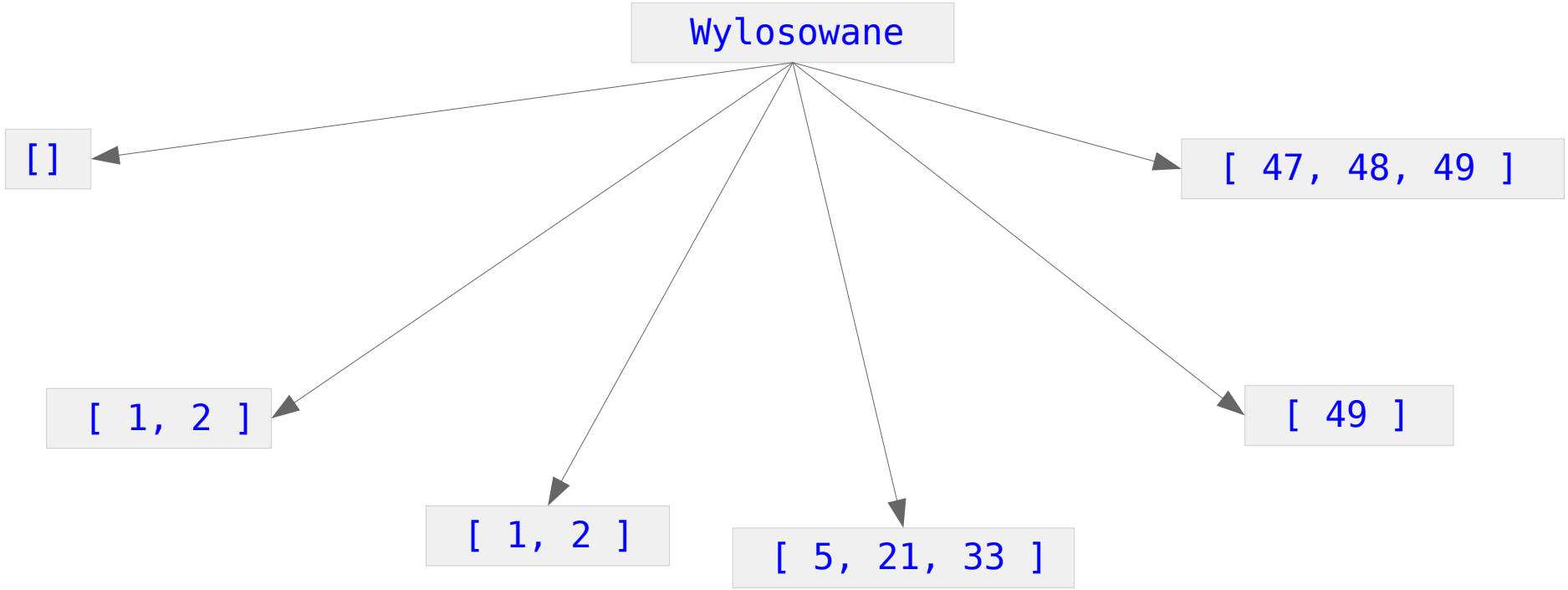
Copyright © Roman Simiński

Definiowanie zbiorów

Aby zdefiniować zbiór o elementach typu liczba całkowita, należy wprowadzić typ okrojony:

```
Var
  Wylosowane : Set Of 1 .. 49;
```

Zmienna Wylosowane może mieć przykładowe wartości:



Copyright © Roman Simiński

Wytypuj szóstkę do „dużego lotka” – wykorzystanie zbiorów

```
Program LosujSzostke;
Const
  (* Losujemy liczby od jeden do wartosci stalej Zakres *)
  Zakres = 49;

Type
  (* Typ okrojony odpowiadajacy puli losowanych liczb   *)
  PulaLiczb = 1 .. Zakres;

Var
  (* Zbior losowanych liczb *)
  Wylosowane    : Set of PulaLiczb;

  (* Tutaj wylosowana liczba, kandydat do losowanej szostki *)
  Liczba        : PulaLiczb;

  (* Licznik losowanych liczb  *)
  Wylosowanych  : Integer;

                                                  (* Ciag dalszy >> *)
```

Copyright © Roman Simiński

Wytypuj szóstkę do „dużego lotka” — wykorzystanie zbiorów

```
Begin
  WriteLn( 'Losuje szostke liczb z 49-ciu.' );

  (* Jeszcze nie wylosowano zadnej liczby, zbior wylosowanych ma byc pusty *)
  Wylosowane := [];

  (* Inicjalizacja generatora liczb pseudolosowych *)
  Randomize;
  (* Zerowanie licznika wylosowanych liczb *)
  Wylosowanych := 0;
  Repeat
    (* Losowanie liczby z zakresu *)
    Liczba := Random( Zakres ) + 1;
    (* Czy czasem juz tej liczby nie wylosowano ? *)
    If Not ( Liczba In Wylosowane ) Then
      Begin
        (* Nie bylo jeszcze takiej liczby, wstawiamy ja do zbioru *)
        Wylosowane := Wylosowane + [ Liczba ];
        (* Zwiekszamy licznik wylosowanych liczb *)
        Wylosowanych := Wylosowanych + 1;
      End;
      (* Powtarzamy az wylosoujemy szostke *)
  Until Wylosowanych = 6;

                                                    (* Ciag dalszy >> *)
```

Copyright © Roman Simiński

Wytypuj szóstkę do „dużego lotka” — wykorzystanie zbiorów

```
  (* Wyswietlenie wylosowanej szostki *)
  WriteLn( 'Wylosowana szostka: ' );

  (* Sprawdz, ktore liczby z zakresu sa w zbiorze wylosowanych *)
  For Liczba := 1 To Zakres Do
    If Liczba In Wylosowane Then (* Jezeli liczba zostala wylosowana *)
      Write( Liczba, ' ' ); (* Wyprowadz liczbe *)

  WriteLn;
  WriteLn( 'Nacisnij Enter by zakonczyc program...' );
  ReadLn;
End.
```

Copyright © Roman Simiński