podstawy i języki programowania

pjp

Podstawy programowania

Część ósma

Tablice znaków i przetwarzanie napisów

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.

Przetwarzanie napisów

- Przetwarzanie napisów to ważny element programowania.
- Napisy mają zmienną długość.
- Jak reprezentować napisy o zmiennej długości?
- Napis to ciąg znaków, zatem najlepsza byłaby reprezentacja w postaci tablicy znaków… .
- Ale tablice mają rozmiar ustalany na etapie kompilacji — jak poradzić sobie z reprezentowaniem napisów o zmiennym rozmiarze?

W standardzie języka Pascal nie przewidziano żadnej metody reprezentowania napisów o zmiennym rozmiarze.

Firma Borland wprowadziła do języka Turbo Pascal typ *String*, służący do reprezentowania krótkich napisów o zmiennym rozmiarze.

Ten typ stał się standardem *de facto*.

Uwaga — ze względu na ograniczenia typu String wprowadzone inne typy dla reprezentacji napisów (np. *AnsiString* i *PChar* w Delphi)

Copyright © Roman Simiński

Typ String — Turbo Pascal i jego następcy i klony

```
Program T1;
Var
  Imie : String;
Begin
  WriteLn;
  Write( 'Wpisz imie: ' );
  ReadLn( Imie );

  WriteLn( 'Wpisales: ', Imie );
  WriteLn( 'Liczba znakow w Twoim imieniu: ', Length( Imie ) );
End.
```

```
Wpisz imie: Ala
Wpisales: Ala
Liczba znakow w Twoim imieniu: 3
```

Jak reprezentowana jest zmienna *Imię*?

Imie	A	l	a	?	?	?	?	?	?	...	?
	1	2	3	4	5	6	7	8	9	...	255

Typ String można interpretować jako *coś podobnego* do tablicy znaków:

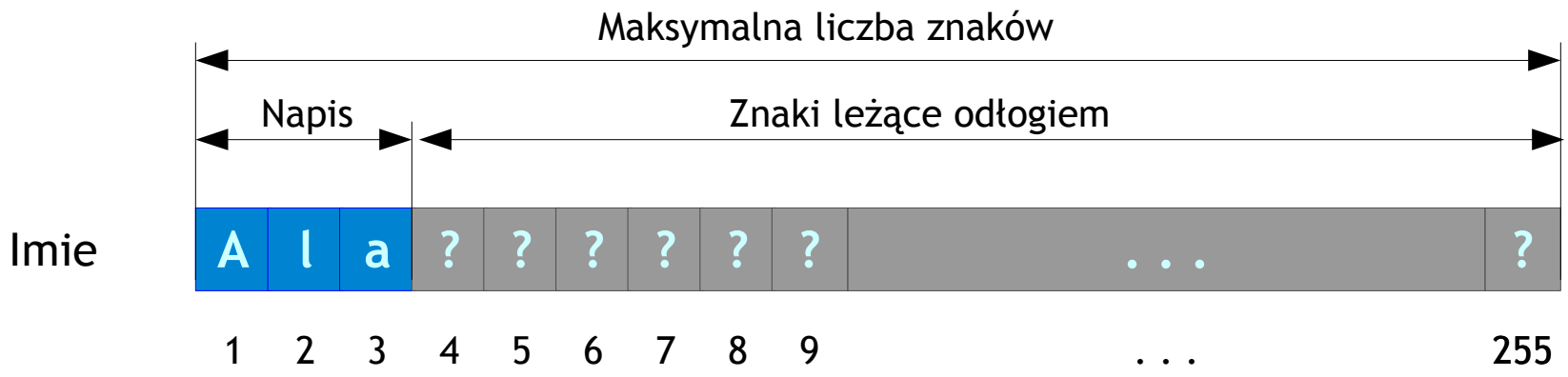
```
Imie : String;
```

⟷

```
Imie : Array[ 1 .. 255 ] Of Char;
```

Copyright © Roman Simiński

Długość napisu w zmiennej typu String




```
Var
  L : Integer;
. . .
L := Length( Imie );
```

L = 3

- Rezultatem funkcji *Length* jest faktyczna liczba znaków przechowywana w zmiennej typu *String*.
- Maksymalna liczba znaków w zmiennej *String* to 255.
- Niewykorzystana część zmiennej typu *String* leży odłogiem i się marnuje.

Copyright © Roman Simiński

Długość napisu w zmiennej typu String, cd...

```
Var
  L : Integer;
. . .
Imie := 'Alina';
L := Length( Imie );
. . .
Imie := 'Kalina';
L := Length( Imie );
. . .
Imie := '';
L := Length( Imie );
```

L = 5

L = 6

L = 0

Skąd funkcja *Length* wie, jaka jest właściwa długość napisu w zmiennej typu *String*?

Copyright © Roman Simiński

Rola ukrytego, 0-ego elementu tablicy

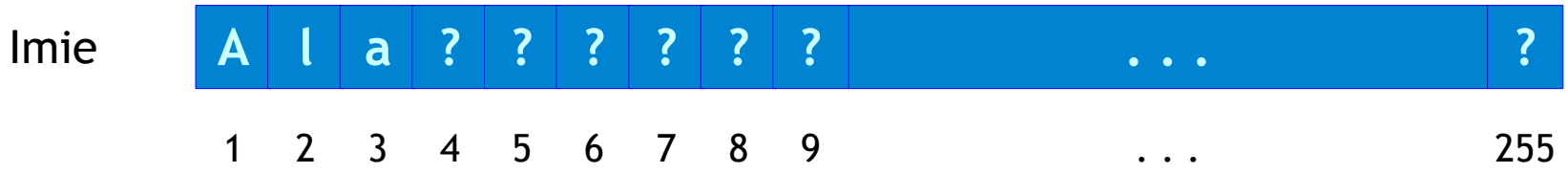
Tak na prawdę, każda zmienna typu *String* posiada niejawny element o indeksie 0.

Do tej pory typ String interpretowaliśmy jako *coś podobnego* do tablicy znaków:

```
Imie : String;
```

⟺

```
Imie : Array[ 1 .. 255 ] Of Char;
```

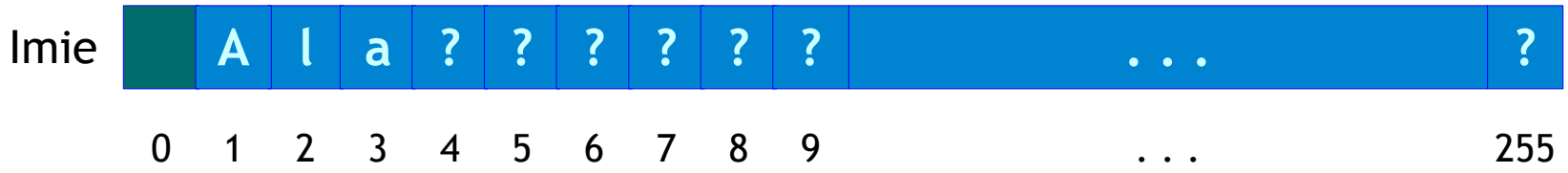


W istocie jest trochę inaczej:

```
Imie : String;
```

⟺

```
Imie : Array[ 0 .. 255 ] Of Char;
```

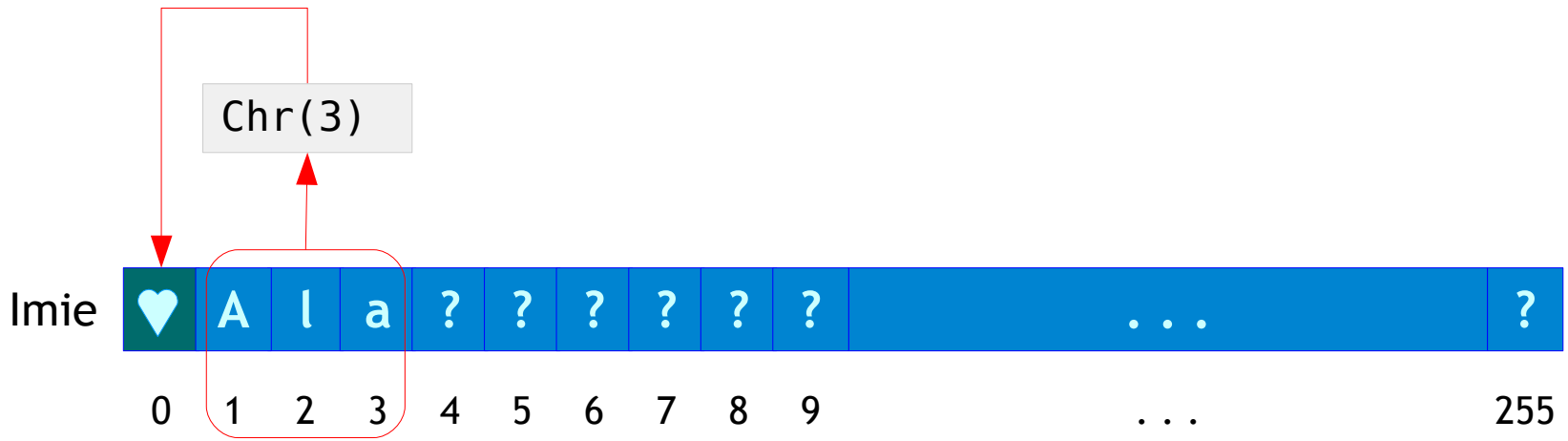


Po co ten dodatkowy, utajniony element zmiennej typu *String*?

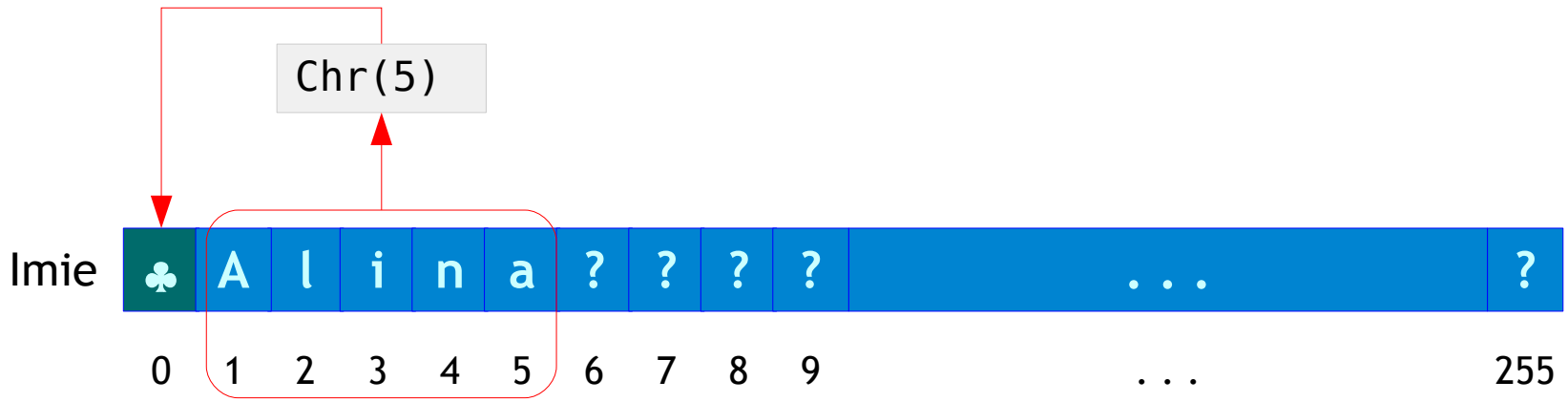
Copyright © Roman Simiński

Rola ukrytego, 0-ego elementu tablicy, cd...

- Znak i indeksie 0 zawiera w sobie zakodowaną liczbę znaków zapisanych w zmiennej typu *String*.
- Dlaczego „zakodowaną”? 0-wy element tablicy jest *znakiem* a długość napisu *liczbą*... nie można pamiętać liczby w zmiennej znakowej... .
- Chyba, ze zastosujemy sztuczkę — w 0-wym elemencie tablicy zapamiętamy *znak* o kodzie odpowiadającym *długości* napisu ;-)



Rola ukrytego, 0-ego elementu tablicy, cd...




```
Var
  L : Integer;
. . .
Imie := 'Alina';
L := Length( Imie );
```

- Kompilatory języka Pascal, obsługujące typ *String*, traktują zmienne tego typu w szczególny sposób, wstawiając do 0-wego elementu zmiennej wartość znakową, odpowiadająca długości napisu.
- We większości operacji — np. *Write*, *WriteLn* — ukryty element informuje, ile znaków liczy napis, a sam ten element jest pomijany.

Copyright © Roman Simiński

Rola ukrytego, 0-ego elementu tablicy, cd...

Skoro zmienną typu *String* można potraktować jako tablicę znaków, czy można zamiast wywołania funkcji *Length* napisać tak:

```
Var
  L : Integer;
. . .
Imie := 'Alina';

L := Length( Imie );

WriteLn( L );
```

```
Var
  L : Integer;
. . .
Imie := 'Alina';

L := Ord( Imie[ 0 ] );

WriteLn( L );
```

5

Można, rzeczywiście funkcja *Length* po prostu sprawdza kod znaku zapisanego w 0-ym elemencie tablicy *Imie*.

Copyright © Roman Simiński

Manipulowanie długością napisu

Czy można sztucznie skrócić długość napisu w zmienney typu *String* poprzez manipulowanie zawartości 0-ego elementu?

```
Var
  S : String;
  L : Integer;
. . .
S := 'Podstawy programowania';

WriteLn( S );

S[ 0 ] := Chr( 8 );

WriteLn( S );

S[ 0 ] := Chr( 0 );

WriteLn( S );
```

```
Podstawy programowania
```

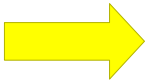
```
Podstawy
```

Uwaga — nie wszędzie typ String jest taki sam!

Typ *String* ma ograniczenia — liczba znaków pojedynczego napisu jest limitowana do 255 znaków. Ponieważ bywa to uciążliwe, w różnych wersjach języka Pascal i w różnych środowiskach RAD wprowadza się własne, mniej lub bardziej ukryte sposoby reprezentacji napisów o zmiennej długości (np. *AnsiString*, *PChar*).

Dlatego lepiej nie odwoływać się bezpośrednio do 0-ego elementu zmiennej typu *String* a używać odpowiednich funkcji przeznaczonych do tego celu. Uwaga — należy uważnie przejrzeć *dokumentację* języka i danego środowiska programistycznego!

```
Var
  S : String;
. . .
S := 'Podstawy programowania';

WriteLn( S );
WriteLn( Ord( S[ 0 ] ) );
S[ 0 ] := Chr( 8 );
WriteLn( S );
WriteLn( Ord( S[ 0 ] ) );
```



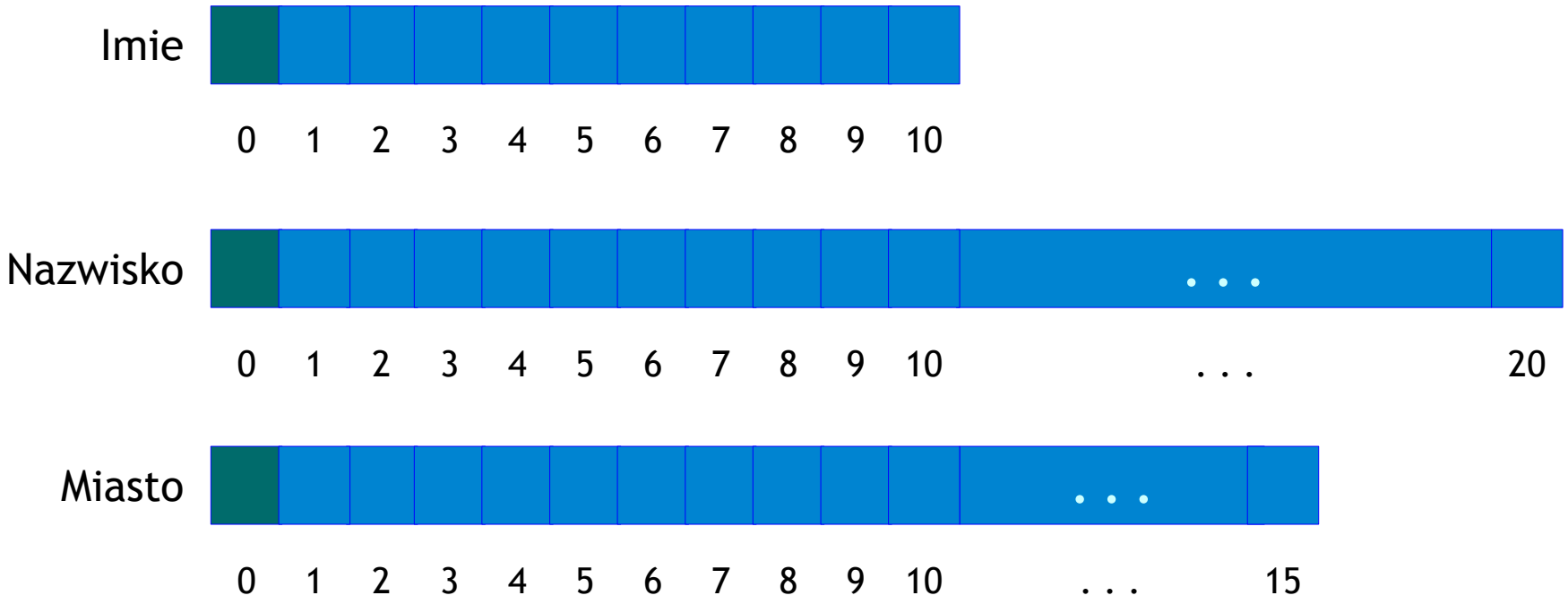
```
Var
  S : String;
. . .
S := 'Podstawy programowania';

WriteLn( S );
WriteLn( Length( S ) );
SetLength( S, 8 );
WriteLn( S );
WriteLn( Length( S ) );
```

Copyright © Roman Simiński

Rozmiar zmiennej typu String można ograniczać

Deklarując zmienną typu String można określić jej maksymalną długość:

```
Var
  Imie     : String[ 10 ];
  Nazwisko : String[ 20 ];
  Miasto   : String[ 15 ];
```



Copyright © Roman Simiński

Użyteczne operacje na zmiennych typu String

Policz ile jest dużych liter w napisie wczytanym z klawiatury:

```
Program LiczbaDuzychLiter;
Var
  LiczDL : Integer; (* Licznik duzych liter         *)
  I      : Integer; (* Zmienna robocza iteracji For *)
  Napis  : String;  (* Przechowuje wczytany napis   *)
Begin
  Write( 'Podaj dowolny napis: ' );
  ReadLn( Napis );

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

  (* Iteracja zliczajaca duze litery *)
  For I := 1 To Length( Napis ) Do
    If Napis[ I ] In [ 'A' .. 'Z' ] Then
      LiczDL := LiczDL + 1;

  (* Wyswietlenie liczby duzych liter *)
  WriteLn( 'Wpisales: ', Napis );
  WriteLn( 'Liczba dużych liter: ', LiczDL );
  WriteLn( 'Nacisnij Enter by zakonczyc...' );
  ReadLn;
End.
```

Copyright © Roman Simiński

Użyteczne operacje na zmiennych typu String

Zamień każdą spacje występującą we wczytanym napisie na znak podkreślenia.

```
Program ZamienSpacjeNaPodkreslenie;
Var
  I      : Integer; (* Zmienna robocza iteracji For *)
  Napis  : String;  (* Przechowuje wczytany napis   *)
Begin
  Write( 'Podaj dowolny napis: ' );
  ReadLn( Napis );


  (* Iteracja zamieniajaca spacje na podkreslenie *)
  For I := 1 To Length( Napis ) Do
    If Napis[ I ] = ' ' Then
      Napis[ I ] := '_';

  (* Wyswietlenie napisu po zamianie *)
  WriteLn( 'Po zamianie: ', Napis );

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

Copyright © Roman Simiński

Użyteczne operacje na zmiennych typu String

Zamień każdą dużą literę występującą we wczytanym napisie na małą.

```
Program ZamienDuzeNaMale;
Var
  I      : Integer; (* Zmienna robocza iteracji For *)
  Napis  : String;  (* Przechowuje wczytany napis   *)
Begin
  Write( 'Podaj dowolny napis: ' );
  ReadLn( Napis );


  (* Iteracja zamieniajac litery duze na male *)
  For I := 1 To Length( Napis ) Do
    If Napis[ I ] In [ 'A' .. 'Z' ] Then
      Napis[ I ] := Chr( Ord( Napis[ I ] ) + 32 );

  (* Wyswietlenie napisu po zamianie *)
  WriteLn( 'Po zamianie: ', Napis );

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

Copyright © Roman Simiński

Użyteczne operacje na zmiennych typu String

Zamień każdą małą literę występującą we wczytanym napisie na dużą.

```
Program ZamienMaleNaDuze;
Var
  I      : Integer; (* Zmienna robocza iteracji For *)
  Napis  : String;  (* Przechowuje wczytany napis   *)
Begin
  Write( 'Podaj dowolny napis: ' );
  ReadLn( Napis );


  (* Iteracja zamieniajaca litery male na duze *)
  For I := 1 To Length( Napis ) Do
    If Napis[ I ] In [ 'a' .. 'z' ] Then
      Napis[ I ] := Chr( Ord( Napis[ I ] ) - 32 );

  (* Wyswietlenie napisu po zamianie *)
  WriteLn( 'Po zamianie: ', Napis );

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

Copyright © Roman Simiński

Można wykorzystać iterację *While* zamiast *For*...

```
Program ZamienMaleNaDuzeWhile;
Var
  I      : Integer; (* Zmienna robocza iteracji For *)
  Napis  : String;  (* Przechowuje wczytany napis   *)
Begin
  Write( 'Podaj dowolny napis: ' );
  ReadLn( Napis );

  (* Iteracja zamieniajaca litery male na duze *)
  I := 1;
  While I <= Length( Napis ) Do
    Begin
      If Napis[ I ] In [ 'a' .. 'z' ] Then
        Napis[ I ] := Chr( Ord( Napis[ I ] ) - 32 );
      I := I + 1;
    End;

  (* Wyswietlenie napisu po zamianie *)
  WriteLn( 'Po zamianie: ', Napis );

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

Copyright © Roman Simiński

Wtedy można zamienić np. co drugi znak na dużą literę

```
Program ZamienMaleNaDuzeWhileCoDrugi;
Var
  I      : Integer; (* Zmienna robocza iteracji For *)
  Napis  : String;  (* Przechowuje wczytany napis   *)
Begin
  Write( 'Podaj dowolny napis: ' );
  ReadLn( Napis );

  (* Iteracja zamieniajaca litery male na duze *)
  I := 1;
  While I <= Length( Napis ) Do
    Begin
      If Napis[ I ] In [ 'a' .. 'z' ] Then
        Napis[ I ] := Chr( Ord( Napis[ I ] ) - 32 );
      I := I + 2;
    End;

  (* Wyswietlenie napisu po zamianie *)
  WriteLn( 'Po zamianie: ', Napis );

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

Copyright © Roman Simiński

Uwaga, wywołanie funkcji *Length* może trwać

```
Program ZamienMaleNaDuzeWhile;
Var
  I      : Integer; (* Zmienna robocza iteracji For *)
  Napis  : String;  (* Przechowuje wczytany napis   *)
Begin
  Write( 'Podaj dowolny napis: ' );
  ReadLn( Napis );

  (* Iteracja zamieniajaca litery male na duze *)
  I := 1;
  While I <= Length( Napis ) Do
    Begin
      If Napis[ I ] In [ 'a' .. 'z' ] Then
        Napis[ I ] := Chr( Ord( Napis[ I ] ) - 32 );
      I := I + 1;
    End;

  (* Wyswietlenie napisu po zamianie *)
  WriteLn( 'Po zamianie: ', Napis );

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

Wywołanie funkcji *Length* tyle razy, ile wynosi liczba znaków zmiennej *Napis*

Copyright © Roman Simiński

Wyznacz długość napisu, zapamiętaj w zmiennej, porównuj ze zmienną

```
Program ZamienMaleNaDuzeWhile;
Var
  I       : Integer; (* Zmienna robocza iteracji For              *)
  Napis   : String;  (* Przechowuje wczytany napis                *)
  Dlugosc : Integer; (* Zmienna do zapamiętania dlugosci napisu   *)
Begin
  Write( 'Podaj dowolny napis: ' );
  ReadLn( Napis );

  (* Iteracja zamieniajaca litery male na duze *)
  Dlugosc := Length( Napis );
  I := 1;
  While I <= Dlugosc Do
    Begin
      If Napis[ I ] In [ 'a' .. 'z' ] Then
        Napis[ I ] := Chr( Ord( Napis[ I ] ) - 32 );
      I := I + 1;
    End;

  (* Wyswietlenie napisu po zamianie *)
  WriteLn( 'Po zamianie: ', Napis );

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

Porównanie ze zmienną jest szybsze niż porównanie z rezultatem wywoływanej funkcji

Copyright © Roman Simiński

Czy warto robić coś takiego dla iteracji *For*?

```
. . .
For I := 1 To Length( Napis ) Do
  If Napis[ I ] In [ 'a' .. 'z' ] Then
    Napis[ I ] := Chr( Ord( Napis[ I ] ) - 32 );
. . .
```

- Wartość *początkowa* i *końcowa* dla zmiennej sterującej iteracji *For* jest wyznaczana raz, przed pierwszym jej wykonaniem.
- Zatem w powyższym fragmencie programu funkcja *Length* powinna zostać wywołana tylko raz, a jej wartość zapamiętana dla iteracji *For*.
- W tym przypadku użycie dodatkowej zmiennej niewiele pomoże, ale uwaga — jeżeli długość napisu jest wykorzystywana więcej iż raz — dodatkowa zmienna wyraźnie pomoże.

```
. . .
Dlugosc := Length( Napis );
For I := 1 To Dlugosc Do
  If Napis[ I ] In [ 'a' .. 'z' ] Then
    Napis[ I ] := Chr( Ord( Napis[ I ] ) - 32 );
. . .
```

Copyright © Roman Simiński

Użyteczne operacje na zmiennych typu String, cd...

Wczytaj imię — sprowadź do formatu Xxxxxx

```
Program WczytajImie;
Var
  I      : Integer; (* Zmienna robocza iteracji For *)
  Imie   : String;  (* Przechowuje wczytany napis   *)
  Dlug   : Integer; (* Przechowuje dlugosc napisu   *)
Begin
  Write( 'Podaj imie: ' );
  ReadLn( Imie );

  Dlug := Length( Imie );
  (* Wszystkie litery oprocz 1-szej mają być male *)
  For I := 2 To Dlug Do
    If Imie[ I ] In [ 'A' .. 'Z' ] Then
      Imie[ I ] := Chr( Ord( Imie[ I ] ) + 32 );
  (* Jezeli napis nie jest pusty, podmien pierwsza litere na duza *)
  If ( Dlug > 0 ) And ( Imie[ 1 ] In [ 'a' .. 'z' ] ) Then
    Imie[ 1 ] := Chr( Ord( Imie[ 1 ] ) - 32 );

  (* Wyswietlenie po normalizacji *)
  WriteLn( 'Po zamianie: ', Imie );
  WriteLn( 'Nacisnij Enter by zakonczyc...' );
  ReadLn;
End.
```

Copyright © Roman Simiński

Użyteczne operacje na zmiennych typu String, cd...

Wczytaj imię — sprowadź do formatu Xxxxxx

```
Program WczytajImie;
Var
  I      : Integer; (* Zmienna robocza iteracji For *)
  Imie   : String;  (* Przechowuje wczytany napis   *)
  Dlug   : Integer; (* Przechowuje dlugosc napisu   *)
Begin
  Write( 'Podaj imie: ' );
  ReadLn( Imie );

  Dlug := Length( Imie );
  (* Wszystkie litery oprocz 1-szej mają być male *)
  For I := 2 To Dlug Do
    If Imie[ I ] In [ 'A' .. 'Z' ] Then
      Imie[ I ] := Chr( Ord( Imie[ I ] ) + 32 );
  (* Jezeli napis nie jest pusty, podmien pierwsza litere na duza *)
  If ( Dlug > 0 ) And ( Imie[ 1 ] In [ 'a' .. 'z' ] ) Then
    Imie[ 1 ] := Chr( Ord( Imie[ 1 ] ) - 32 );

  (* Wyswietlenie po normalizacji *)
  WriteLn( 'Po zamianie: ', Imie );
  WriteLn( 'Nacisnij Enter by zakonczyc...' );
  ReadLn;
End.
```

Użyj dodatkowej zmiennej dla liczby znaków w napisie gdy jest ona wielokrotnie wykorzystywana w programie

Copyright © Roman Simiński

Użyteczne operacje na zmiennych typu String, cd...

Wczytaj imię — sprawdź poprawność i sprowadź do formatu Xxxxxx

```
Program WczytajImieZKontrolaPoprawnosci;
Var
  I      : Integer; (* Zmienna robocza iteracji For *)
  Imie   : String;  (* Przechowuje wczytany napis   *)
  Dlug   : Integer; (* Przechowuje dlugosc napisu   *)
  ImieOK : Boolean;
Begin
  Write( 'Podaj imie (tylko litery) : ' );
  ReadLn( Imie );

  ImieOK := True; (* Zakladam, ze imie jest poprawne *)
  Dlug := Length( Imie );

  (* Wszystkie znaki musza byc literami *)
  For I := 1 To Dlug Do
    If Not ( Imie[ I ] In [ 'A' .. 'Z', 'a'.. 'z' ] ) Then
      ImieOK := False;

  If Not ImieOK  Then
    WriteLn( 'Wprowadzone imie nie jest poprawne' )
  Else
    (* Tu to, co na poprzednim slajdzie *)
End.
```

Copyright © Roman Simiński

Użyteczne operacje na zmiennych typu String, cd...

Nie zawsze iteracja *For* jest optymalna (zawsze *Dlugosc* przebiegów):

```
. . .
ImieOK := True; (* Zakladam, ze imie jest poprawne *)

(* Wszystkie znaki musza byc literami *)
For I := 1 To Length( Imie ) Do
  If Not ( Imie[ I ] In [ 'A' .. 'Z', 'a'.. 'z' ] ) Then
    ImieOK := False;
. . .
```

W przypadku błędnego imienia iteracja *While* ma szansę zakończyć się wcześniej:

```
. . .
ImieOK := True; (* Zakladam, ze imie jest poprawne *)
Dlug := Length( Imie );
I := 1;
(* Wszystkie znaki musza byc literami *)
While ( I <= Dlug ) And ImieOK Do
  Begin
    If Not ( Imie[ I ] In [ 'A' .. 'Z', 'a'.. 'z' ] ) Then
      ImieOK := False;
    I := I + 1;
  End;
. . .
```

Copyright © Roman Simiński

Weryfikacja poprawności wprowadzonej liczby całkowitej

```
Program WczytajLiczbeZKontrolaPoprawnosci;
Var
  I        : Integer; (* Zmienna robocza iteracji For       *)
  Napis    : String;  (* Przechowuje wczytany napis         *)
  Dlug     : Integer; (* Przechowuje dlugosc napisu         *)
  LiczbaOK : Boolean; (* Czy wczytana liczba jest poprawna? *)
  Liczba   : Integer; (* Zmienna numeryczna                 *)
Begin
  Write( 'Podaj liczbe calkowita: ' );
  ReadLn( Napis );

  Dlug := Length( Napis );
  (* Tylko cyfry, na pierwszej pozycji moĭe byc teĭ znak + lub - *)
  LiczbaOK := ( Dlug > 0 ) And ( Napis[ 1 ] In [ '0' .. '9', '+', '-' ] );
  I := 2;
  While ( I <= Dlug ) And LiczbaOK Do
    Begin
      If Not ( Napis[ I ] In [ '0'.. '9' ] ) Then
        LiczbaOK := False;
      I := I + 1;
    End;
                                                    (* Ciag dalszy >> *)
```

Copyright © Roman Simiński

Weryfikacja poprawności wprowadzonej liczby całkowitej

```
  If Not LiczbaOK Then
    WriteLn( 'Wprowadzona liczba nie jest poprawna' )
  Else
    Begin
      Val( Napis, Liczba );
      WriteLn( 'Wprowadzono liczbe: ', Liczba );
    End;
  WriteLn( 'Nacisnij Enter by zakonczyc...' );
  ReadLn;
End.
```

Zamień ciąg cyfr zapisany w zmiennej *Napis* na liczbę i wstaw do zmiennej *Liczba*

```
Val( Napis , Liczba );
```

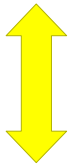
Zmienna typu String

Zmienna typu Integer

Copyright © Roman Simiński

Weryfikacja poprawności wprowadzonej liczby całkowitej – poprawka

```
(* Tylko cyfry, na pierwszej pozycji moĭe byc teĭ znak + lub - *)
LiczbaOK := ( Dlug > 0 ) And ( Napis[ 1 ] In [ '0' .. '9', '+', '-' ] );
I := 2;
While ( I <= Dlug ) And LiczbaOK Do
  Begin
    If Not ( Napis[ I ] In [ '0'.. '9' ] ) Then
      LiczbaOK := False;
    I := I + 1;
  End;
```



```
(* Tylko cyfry, na pierwszej pozycji moĭe byc teĭ znak + lub - *)
LiczbaOK := ( Dlug > 0 ) And ( Napis[ 1 ] In [ '0' .. '9', '+', '-' ] );
I := 2;
While ( I <= Dlug ) And LiczbaOK Do
  Begin
    LiczbaOK := ( Napis[ I ] In [ '0'.. '9' ] );
    I := I + 1;
  End;
```

Copyright © Roman Simiński

Weryfikacja poprawności wprowadzonej liczby rzeczywistej

- Przedstawiony algorytm walidacji liczby można dostosować również dla liczb typu *Real*.
- W tym przypadku sprawa się *komplikuje* ze względu na *kropkę dziesiętną* i potencjalny zapis liczby w *postaci wykładniczej*.
- Można jednak skorzystać z trzeciego, opcjonalnego parametru funkcji *Val*.
- Po wykonaniu funkcji *Val*, *trzeci parametr* otrzymuje wartość 0 jeżeli przekazany tej funkcji napis zawierał poprawna liczbę.
- Jeżeli w napisie wystąpił *niedozwolony znak*, *trzeci parametr* otrzymuje wartość *numeru błędnej pozycji* w napisie.

```
Val( Napis , Liczba , Blad );
```

Napis → Zmienna typu String

Liczba → Zmienna typu Integer

Blad → Numer błędnej pozycji

Copyright © Roman Simiński

Weryfikacja poprawności wprowadzonej liczby rzeczywistej

```
Program WczytajLiczbeRealZKontrolaPoprawnosci;
Var
  Napis    : String;  (* Przechowuje wczytany napis         *)
  Blad     : Integer; (* Numer blednego znaku w napisie     *)
  Liczba   : Real;    (* Zmienna numeryczna                 *)
Begin

  Repeat
    Write( 'Podaj liczbe: ' );
    ReadLn( Napis );

    Val( Napis, Liczba, Blad );
    If Blad <> 0 Then
      WriteLn( 'wprowadzona liczba jest nieprawidlowa' );
  Until Blad = 0;


  WriteLn( 'Wprowadzono liczbe: ', Liczba : 0 : 2 );

  . . .

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

Copyright © Roman Simiński