STUDIA INFORMATICA 2002
Volume 23 Number 4 (51)

Damian GRUND
Politechnika Śląska, Instytut Informatyki

BIBLIOTEKA FUNKCJI DOSTĘPU DO DANYCH SŁOWNIKA SYNTAKTYCZNEGO CZASOWNIKÓW POLSKICH

Streszczenie. W artykule opisano bibliotekę zawierającą funkcje umożliwiające dostęp oraz analizę danych zawartych w komputerowej implementacji słownika syntaktyczno-generatywnego czasowników polskich. Przedstawiono algorytm dostępu do danych zawartych w bazie, opisano sposób wykorzystania dekodera oraz translatora schematów, określono zasady kodowania poszczególnych elementów schematów oraz przedstawiono możliwości zastosowania biblioteki.

ACCESS FUNCTIONS LIBRARY FOR SYNTATIC DICTIONARY OF POLISH VERBS

Summary. The article presents a software library providing access and analysis functions for data contained in digital version of syntactic dictionary of Polish verbs. An algorithm of data access is presented, methods of application the sentence scheme decoder and translator are described. The rules of scheme encoding are briefly sketched and possible applications of the library are enumerated.

1. Wstęp

Znaczna część informacji docierających do człowieka w dzisiejszych czasach jest przekazywana za pomocą języka naturalnego, dlatego istnieje konieczność ciągłego ulepszania metod jego przetwarzania. Języki zarówno naturalne, jak i formalne analizuje się na podstawie wcześniej zdefiniowanych gramatyk. W przypadku gramatyki opisującej język naturalny, mamy do czynienia z wieloma słownikami o dużej objętości, dlatego niezbędne są narzędzia umożliwiające szybkie dotarcie do wyszukiwanych informacji. Jednym z takich słowników jest słownik zawierający informacje syntaktyczne o czasownikach polskich [1], który został zaimplementowany w postaci komputerowej bazy danych Semsyn [2].

Dla wspomnianej bazy danych został zaimplementowany interfejs umożliwiający odczyt wszystkich niezbędnych danych związanych z czasownikiem. Odczytane dane pozwalają na pełne odtworzenie zawartości słownika w wersji drukowanej, ale ze względu na format zapisu nie nadają się bezpośrednio do wykorzystania przez takiego użytkownika, jakim jest programista lub badacz zainteresowany komputerowym przetwarzaniem tekstu w języku naturalnym. Potrzeba takiego dostępu pojawiła się w związku z pracami nad translatorem tekstów z języka polskiego na język migowy. Niniejsza praca poświęcona jest opisowi uniwersalnego narzędzia wspomagającego dostęp do informacji syntaktyczno-generatywnych zawartych w bazie danych Semsyn. Uniwersalność tego narzędzia polega na tym, że umożliwia dostęp do informacji zawartych w słowniku bez konieczności znajomości organizacji bazy danych oraz że daje możliwość wykorzystania w różnych językach programowania, np. Microsoft Visual C, Microsoft Visual Basic, Borland Delphi oraz wielu innych.

2. Opis budowy biblioteki

Jak wyżej wspomniano, baza danych Semsyn została stworzona nie tylko jako materiał informacyjny, ale także z myślą o wykorzystywaniu jej zawartości z poziomu programów użytkownika. Dlatego przy użyciu języka programowania Visual C została zaimplementowana biblioteka o nazwie ssx.ocx. Ma ona postać kontrolki ActiveX [3] i umożliwia uzyskiwanie z elektronicznej postaci słownika syntaktyczno-generatywnego wszystkich zawartych tam informacji, takich jak schematy syntaktyczne związane z danym czasownikiem, znaczenie oraz przykład zastosowania czasownika w danym schemacie, jak też opisy semantyczne poszczególnych elementów wchodzących w skład schematu syntaktycznego.

Na kontrolkę składają się trzy powiązane ze sobą moduły. Pierwszy z nich to moduł bazy umożliwiający odczyt zawartości danych z bazy. Odczyt prowadzi się zgodnie ze strukturą bazy danych, lecz dane te, jak wspomniano, nie nadają się do bezpośredniego wykorzystania.

Schematy syntaktyczne zostały zapisane w bazie w postaci ciągu znaków w oparciu o zbiór znaków i symboli pomocniczych przedstawionych na rys. 2, 3, 4 oraz dokładnie opisanych w literaturze [2]. Taki sposób zapisu jest dla człowieka stosunkowo łatwy do odczytu i zrozumienia, jednak w przypadku przetwarzania komputerowego, w trakcie którego schemat jest analizowany i interpretowany, okazuje się niepraktyczny i mało efektywny. Aby temu zaradzić, w kontrolce wbudowano moduł dekodera, który na podstawie ogólnego schematu syntaktycznego tworzy schematy pochodne. Schematy te zostały zapisane w oparciu o wcześniej określone znaki i symbole pomocnicze. Trzecim modułem jest moduł

translatora, którego zadaniem jest analiza poszczególnych schematów oraz kodowanie do postaci ciągu liczb o długości dwóch bajtów. W trakcie analizy zapamiętywana jest także pozycja elementu w schemacie, co umożliwia odczyt cech semantycznych każdego elementu.

2.1. Moduł bazy

Moduł bazy dostarcza użytkownikowi wielu procedur i funkcji umożliwiających odczyt danych bezpośrednio z bazy. Ich pełna lista została umieszczona w rozdziale trzecim.

Entry	Id
Zachęcać	107

Id	Nr Schem	Nr Desc
107	84	260
107	83	259

Nr Schem	Schema
83	Npn-{{Npacc+{Or,"do"^Npg,"żeby"^S}+(Npi)
84	Npn-Npacc+{"do"^Npg,"żeby"^S}

Nr Desc	Desc	Example
259		Anna bezskutecznie zachęcała syna do podjęcia studiów.
260		Pogoda zachęcała do wyjazdu.

Nr_Desc	Nr_Symb	Abstr	nAbstr	Anim	nAnim	Hum	nHum	Coll	Elm	Fl	Inf	Insti	Instr	Liqu	Mach	Mat	Pars
259	1	Nie	Nie	Nie	Nie	Tak	Nie	Nie	Nie	Nie	Nie	Nie	Nie	Nie	Nie	Nie	Nie
259	2	Nie	Nie	Nie	Nie	Tak	Nie	Nie	Nie	Nie	Nie	Nie	Nie	Nie	Nie	Nie	Nie
259	4	Tak	Nie	Nie	Nie	Nie	Nie	Nie	Nie	Nie	Nie	Nie	Nie	Nie	Nie	Nie	Nie
259	6	Tak	Nie	Nie	Nie	Nie	Nie	Nie	Nie	Nie	Nie	Nie	Nie	Nie	Nie	Nie	Nie
260	1	Tak	Nie	Nie	Nie	Nie	Nie	Nie	Nie	Nie	Nie	Nie	Nie	Nie	Nie	Nie	Nie
260	2	Nie	Nie	Nie	Nie	Tak	Nie	Nie	Nie	Nie	Nie	Nie	Nie	Nie	Nie	Nie	Nie
260	3	Tak	Nie	Nie	Nie	Nie	Nie	Nie	Nie	Nie	Nie	Nie	Nie	Nie	Nie	Nie	Nie

Rys. 1. Sposób rozmieszczenia danych o czasowniku *zachęcać* w bazie
Fig. 1. The method of data distribution for verb *zachęcać* in the database

Do najważniejszych należą: wyznaczanie numeru *Idv* czasownika, liczby *Lsch* schematów syntaktycznych, numeru *Idd* określającego adres opisu czasownika dla określonego schematu, numeru *Ids* określającego położenie schematu w bazie, jak również funkcje umożliwiające pobieranie schematów syntaktycznych, opisów semantycznych, przykładów zastosowania i znaczeń czasowników.

Na rys. 1 przedstawiono sposób zapisu danych w bazie Semsyn dla czasownika *zachęcać*.

Jak widać na rys. 1, czasownik *zachęcać* ma dwa różne schematy syntaktyczne wraz z opisami semantycznymi. Dla każdego schematu w bazie danych został przedstawiony przykład zastosowania, natomiast nie zostały zdefiniowane znaczenia czasownika. Dane o czasowniku odczytujemy w następującej kolejności:

1. Odczytujemy numer *Idv* czasownika, wykorzystując funkcję *GetDbEntryId(Entry)*

   Idv = *GetDbEntryId*(„*zachęcać*").

Dla czasownika *zachęcać* jest on równy 107.

2. Określamy liczbę *Lsch* schematów syntaktycznych związanych z danym czasownikiem za pomocą funkcji *GetDbSchCount*(*Idv*).

   Lsch = *GetDbSchCount*(107)

Dla czasownika *zachęcać* liczba ta wynosi 2.

3. Określamy adresy opisów *Idd* związanych z poszczególnymi schematami funkcją:

   GetDbIdDescPos(*Idv*, *Pos*),

gdzie parametr *Pos* określa numer schematu (liczony od 1).

   Idd1 = *GetDbIdDescPos*(107, 1)

   Idd2 = *GetDbIdDescPos*(107, 2)

Dla czasownika *zachęcać Idd1* jest równe 259, natomiast *Idd2* ma wartość 260.

4. Określamy adresy schematów syntaktycznych Ids, wykorzystując funkcję

   GetDbSchPos(*Idv*, *Pos*).

   Sch1 = *GetDbSchPos*(107, 1)

   Sch2 = *GetDbSchPos*(107, 2)

Dla naszego czasownika *Sch1* ma wartość 83, natomiast *Sch2* 84.

5. Mając dane adresy opisów *Idd1* i *Idd2* możemy odczytać znaczenie i przykład zastosowania dla poszczególnych schematów. Służą do tego funkcje *GetDbExample*(*Idd*) oraz *GetDbMeaning*(*Idd*).

I tak dla czasownika *zachęcać* odczytujemy dane:

   E1 = *GetDbExample*(259)

   M1 = *GetDbMeaning*(259)

   E2 = *GetDbExample*(260)

   M2 = *GetDbMeaning*(260)

Znaczenia dla czasownika *zachęcać* nie zostały zdefiniowane, dlatego zmienne M1 i M2 zawierają puste ciągi znaków, natomiast zmienne E1 i E2 zawierają zawartości pól *Example*.

6. Mając dane adresy schematów możemy je odczytać za pomocą funkcji *GetDbScheme(Ids)*.

Dla naszego czasownika są to:

Sch1 = *GetDbScheme*(83)

Sch2 = *GetDbScheme*(84)

W wyniku otrzymujemy następujące wartości:

Sch1= Npn-{{Npacc+{Or,"do"^Npg,"żeby"^S}+(Npi)

Sch2= Npn-Npacc+{"do"^Npg,"żeby"^S}

7. Aby określić opisy semantyczne poszczególnych elementów schematu, musimy określić liczbę tych elementów. Służy do tego funkcja *GetDbSchemElCountI*(*Ids*). W naszym przypadku:

LE1 = *GetDbSchemElCountI*(83)

LE2 = *GetDbSchemElCountI*(84)

Dla naszego czasownika *LE1* = 6, a *LE2* = 4.

8. Następnie możemy pobrać opisy słowne (o ile takie istnieją). Służy do tego funkcja *GetDbSemDescAtPosS*(Idd, Pos), gdzie parametr Pos określa numer kolejny opisywanego elementu w schemacie. Czasownik *zachęcać* nie ma opisów słownych, dlatego też funkcja zwróci pustą wartość.

9. Każdy element schematu może mieć kilka opisów semantycznych opartych na cechach. Dlatego też najpierw należy sprawdzić, ile ich dany element posiada.

Służy do tego funkcja *GetDbCountSemDescAtPos*(Idd, Pos). Do odczytu cech semantycznych służy funkcja *GetDbSemDescAtPosW*(Idd, Pos, DescPos), gdzie DescPos określa numer kolejny opisu semantycznego danego elementu.

2.2. Moduł dekodera schematów

Moduł dekodera schematów służy do dekodowania schematów zapisanych w bazie do postaci nadającej się do dalszej analizy. I tak np. czasownik *demonstrować* w znaczeniu *pokazywać* ma schemat o postaci:

NPn–Npacc+({Npd,"przeciw"^Npi,"wobec"^NPg})

Został on zdekodowany i na jego podstawie powstały schematy:

NP_N–NP_{Acc}

NP_N–NP_{Acc}+NP_D

NP_N–NP_{Acc}+przeciw^NP_I

NP_N–NP_{Acc}+wobec^NP_G

W module następuje także powiązanie poszczególnych elementów schematów z ich opisami semantycznymi. Polega ono na wstawianiu za danym elementem w nawiasie klamrowym jego numeru kolejnego w schemacie bazowym. I tak powyższe schematy zostają uzupełnione w następujący sposób.

$NP_N[1]$–$NP_{Acc}[2]$
$NP_N[1]$–$NP_{Acc}[2]$+$NP_D[3]$
$NP_N[1]$–$NP_{Acc}[2]$+przeciw^$NP_I[4]$
$NP_N[1]$–$NP_{Acc}[2]$+wobec^$NP_G[5]$

Aby uruchomić dekoder, należy wywołać funkcję:

RunDec(CString Scheme, BOOL Sem),

gdzie parametr Scheme określa analizowany schemat, natomiast wartość logiczna parametru *Sem* informuje, czy mają zostać zapamiętane pozycje niezbędne do określenia cech semantycznych. Wyniki operacji są zapamiętywane wewnątrz obiektu dekodera i aby je pobrać należy wywołać funkcje: *GetDecSchemCount()* oraz *GetDecSchem(int Pos).* Pierwsza zwraca liczbę wygenerowanych schematów, a druga schemat znajdujący się na danej pozycji.

2.3. Moduł translatora schematów

W słowniku [1] schematy syntaktyczne zostały zapisane zgodnie ze zdefiniowaną notacją, tj. w postaci ciągów znaków. Taki zapis jest w miarę przejrzysty oraz łatwo modyfikowalny, jednak analiza komputerowa takich ciągów jest bardzo utrudniona i kosztowna czasowo. Użytkownik musiałby analizować schemat znak po znaku, łączyć znalezione znaki w symbole i dopiero później zajmować się właściwą analizą tekstu. Aby temu zaradzić, zaimplementowano translator kodu. Jest to narzędzie pozwalające zapisać schematy syntaktyczne nie jako ciąg symboli, ale w postaci tablicy zawierającej pozycjonowane ciągi wartości bitowych. Jeżeli w schemacie występuje tekst, to jest on zapisywany w dodatkowej tablicy. Rozwiązanie takie przyspiesza operacje związane z analizą schematu, gdyż dane są uporządkowane i zapisane w sposób formalny. Aby schemat można było zakodować, musi zostać poddany wstępnej analizie, polegającej na redukcji nawiasów oznaczających alternatywę i koniunkcję.

2.3.1. Kodowanie

Każdy ciąg pozycjonowanych wartości bitowych opisujący dany element schematu został zakodowany w postaci słowa o długości dwóch bajtów.

W pierwszym ciągu o długości dwóch bitów została zawarta informacja o tym, czy dany element schematu jest opisywany czasownikiem, innym elementem schematu określonym

symboliczne, czy też tekstem. Wymienione informacje zostały zakodowane w następujący sposób:

H H - opisywany czasownik
L H - zdefiniowany element schematu
H L - zwykły tekst

Drugi ciąg, także o długości dwóch bitów, informuje, w jakiej liczbie występuje opisywany element.

H L - liczba pojedyncza
L H - liczba mnoga
L L - nie określono liczby

Trzeci ciąg o długości jednego bitu zawiera informacje o znaku łączącym dany element schematu z elementem poprzednim. Stan L informuje, że występuje łączenie bez implikacji szyku w zdaniu lub opisywany element jest pierwszym elementem schematu, natomiast stan H oznacza konkatenację.

W czwartym ciągu o długości czterech bitów zostały zawarte podstawowe informacje o danym elemencie schematu, np. czy jest to fraza nominalna, korelat zaimkowy czy też mowa bezpośrednia. Informacje zostały zakodowane w następujący sposób przedstawiony na rys. 2.

L L L L	-	OR	(mowa bezpośrednia)
L L L H	-	Ip	(fraza bezokolicznikowa)
L L H L	-	K	(partykuła zapytajna)
L L H H	-	S	(zdanie)
L H L L	-	Adv	(przysłówek)
L H L H	-	Adj	(przymiotnik)
H L L L	-	Ts	(korelat zaimkowy)
H H L L	-	NP	(fraza nominalna)

Rys. 2. Sposób kodowania podstawowych informacji o elemencie schematu
Fig. 2. The method of scheme encoding in cases of basic elements

Gdy opisywany element jest frazą nominalną lub korelatem zaimkowym, to w piątym ciągu o długości trzech bitów został zakodowany jego przypadek gramatyczny, natomiast w przypadku czasownika będą w przyszłości zawierać podstawowe informacje o tym czasowniku, np. o jego stronie biernej oraz o tym, czy jest dokonany czy nie. Przypadki gramatyczne zostały zapisane zgodnie z notacją przedstawioną na rys. 3.

L L L	-	brak określonego przypadku	
L L H	-	N	(mianownik)
L H L	-	G	(dopełniacz)
L H H	-	D	(celownik)
H L L	-	Acc	(biernik)
H L H	-	I	(narzędnik)
H H L	-	L	(miejscownik)

Rys. 3. Sposób kodowania przypadków gramatycznych elementów schematu
Fig. 3. The method of grammar cases encoding for sentence schemes

Gdy opisywany element jest frazą nominalną lub przysłówkiem, to szósty ciąg o długości czterech bitów zawiera opis dodatkowych cech tego elementu. Są to:

L L L L	-	brak dodatkowych cech	
L L L H	-	Abl	(oddalanie się)
L L H L	-	Adl	(przybliżanie się)
L L H H	-	Akc	(okoliczność towarzysząca)
L H L L	-	Caus	(przyczyna)
L H L H	-	Cond	(warunek)
L H H L	-	Dest	(cel)
L H H H	-	Grad	(stopień i miara)
H L L L	-	Loc	(miejsce)
H L L H	-	Mod	(sposób)
H L H L	-	Perl	(miejsce, przez które odbywa się ruch)
H L H H	-	Td	(czas trwania)
H H L L	-	Tp	(granica czasowa)
H H L H	-	Cons	(względność)

Rys. 4. Sposób kodowania dodatkowych cech fraz nominalnych i przysłówków
Fig. 4. The method of additional feature encoding for noun groups and adverbs

W przypadku gdy opisywany element jest ciągiem znaków, to w szóstym ciągu jest zapamiętywana jego pozycja w dodatkowej tablicy.

2.3.2. Odczyt danych

W pierwszej kolejności należy sprawdzić bity pierwszego ciągu. Jeżeli analizowanym elementem jest czasownik, to wartości na pozostałych bitach nie mają znaczenia, jeżeli jest to ciąg znaków, czyli tekst, to należy odczytać jego pozycję w tabeli pomocniczej, która została zakodowana w szóstym ciągu. W przypadku gdy analizowany element jest tak zwanym

zdefiniowanym symbolem schematu, czyli określonym w drukowanej wersji słownika [1], to na podstawie zawartości ciągu czwartego można określić rodzaj elementu zgodnie z rys. 2. Jeżeli najstarszy bit ciągu ma wartość logiczną H, oznacza to, że opisywany element może mieć zdefiniowany przypadek gramatyczny, który można określić po odczytaniu wartości bitów z ciągu piątego. W przypadku gdy drugi bit ciągu czwartego ma wartość logiczną H, oznacza to, że dany element może mieć dokładniejszy opis. Opis ten zakodowano na pozycjach ciągu szóstego. Oprócz tego odczytując dane z ciągu drugiego można określić liczbę, w jakiej występuje dany element, natomiast ciąg trzeci zawiera informacje o sposobie łączenia danego elementu z elementem poprzedzającym.

2.3.3. Przykład

W wyniku wstępnej analizy schematu:

Npn–{Npacc+{"za"^Npacc,"za"^Tsacc"że"^S},Npacc}

otrzymano między innymi schemat:

Npn–Npacc+"za"^Tsacc"że"^S

Schemat ten zawiera siedem elementów podlegających opisowi, a są to:

Fraza nominalna w przypadku gramatycznym N
Opisywany czasownik
Fraza nominalna w przypadku gramatycznym Acc
Tekst „za”
Korelat zaimkowy w przypadku gramatycznym Acc
Tekst „że”
Zdanie

Postać, w jakiej schemat ten został zakodowany, przedstawia tab. 1 i 2.

Tabela 1

Zawartość tabeli głównej

Opisywany element	Wartości na poszczególnych pozycjach															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Npn	L	H	L	L	L	H	H	L	L	L	L	H	L	L	L	L
-	H	H	L	L	L	L	L	L	L	L	L	L	L	L	L	L
Npacc	L	H	L	L	L	H	H	L	L	H	L	L	L	L	L	L
„za”	H	L	L	L	L	L	L	L	L	L	L	L	L	L	L	H
^Tsacc	L	H	L	L	H	H	L	L	L	H	L	L	L	L	L	L
„że”	H	L	L	L	L	L	L	L	L	L	L	L	L	L	H	L
S	L	H	L	L	H	L	L	H	H	L	L	L	L	L	L	L

Tabela 2

Zawartość tabeli pomocniczej

Adres	Opis
1	za
2	że

3. Wykaz dostępnych funkcji

Poniżej znajduje się zestawienie wszystkich funkcji udostępnionych przez kontrolkę ActiveX. Typy argumentów oraz zwracanych przez funkcje wartości są zgodne ze specyfikacją firmy Microsoft dla środowiska programistycznego Visual C++ i zostały opisane w literaturze [3]. Funkcje umieszczono w trzech tabelach w zależności od tego, do którego modułu należą.

Tabela 3

Zestawienie funkcji związanych z modułem bazy danych

Nazwa	Typ wyniku	Opis działania	Parametry
OpenDb	void	otwarcie baz danych	brak
CloseDb	void	zamknięcie baz danych	brak
GetDbEntryCount	long	Zwraca liczbę czasowników zapisanych w bazie.	brak
GetDbEntry	CString	Zwraca czasownik o numerze Id.	long Id
GetDbEntryId	long	Zwraca numer Id czasownika Entry.	CString Entry
GetDbSchCount	int	Zwraca liczbę schematów dla danego czasownika.	long Id
GetDbIdDescPos	long	Zwraca numer opisu czasownika o numerze Id dla schematu o numerze kolejnym Pos.	long Id, int Pos
GetDbSchPos	long	Zwraca numer schematu w tabeli t_schem (gdy brak schematu zwraca 0), parametr Pos określa numer kolejny schemat danego czasownika.	long Id, int Pos
GetDbExample	CString	Zwraca przykład na podstawie numeru opisu.	long Id
GetDbMeaning	CString	Zwraca znaczenie czasownika na podstawie numeru opisu.	long Id

cd. tab. 3

GetDbScheme	CString	pobranie schematu na podstawie jego numeru Id	long Id
GetDbSchemId	long	pobranie numeru Id schematu Scheme	CString Scheme
GetDbSchemElCountI	int	Zwraca liczbę elementów wg numeracji w bazie.	long Id
GetDbSchemElCountS	int	Zwraca liczbę elementów wg numeracji w bazie.	CString Scheme
GetDbSchemElAtPosI	CString	Zwraca element o numerze kolejnym Pos ze schematu o numerze IdSchem wg numeracji w bazie.	long Id, int Pos
GetDbSchemElAtPosS	CString	Zwraca element o numerze kolejnym Pos ze schematu Scheme wg numeracji w bazie.	CString Scheme, int Pos
GetDbCountSemDescAtPos	int	Zwraca liczbę opisów semantycznych dla danego opisu schematu syntaktycznego dla elementu o numerze kolejnym Pos.	long Id, int Pos
GetDbSemDescAtPosW	unsigned short	Zwraca opis semantyczny w postaci liczby o dł. 2 bajtów. Pos określa pozycję elementu w schemacie, DescPos określa numer kolejny opisu semantycznego danego elementu.	long Id, int Pos, int DescPos
GetDbSemDescAtPosS	CString	Zwraca słowny opis elementu schematu na pozycji Pos.	long Id, int Pos

Tabela 4

Zestawienie funkcji związanych z modułem dekodera

Nazwa	Typ wyniku	Opis działania	Parametry
RunDec	void	Uruchamianie dekodera, parametr Scheme określa analizowany schemat, zmienna Sem określa czy w schematach wyjściowych mają znajdować się informacje o cechach semantycznych poszczególnych elementów.	CString Scheme, BOOL Sem
GetDecSchemCount	int	Zwraca liczbę wygenerowanych schematów.	brak
GetDecSchem	CString	pobranie wygenerowanego schematu o numerze kolejnym Pos	int Pos
ResetDec	void	kasowanie zawartości tablic dekodera	brak

Tabela 5

Zestawienie funkcji związanych z modułem translatora

Nazwa	Typ wyniku	Opis działania	Parametry
RunTr	void	uruchomienie translatora	brak
ResetTr	void	kasowanie translatora	brak
RunTrSgElem	WORD	translacja pojedynczego elementu	CString Elem
GetTrVerbPos	int	Zwraca pozycję, jaką zajmuje czasownik na liście elementów.	brak
GetTrAbsVerbPos	int	Zwraca pozycję czasownika w schemacie.	brak
GetTrKindOfObjectAtPos	int	Zwraca rodzaj elementu na danej pozycji (1-czas, 2-elem, 3-string, -1-błąd).	int Pos
GetTrElementCount	int	Zwraca liczbę elementów schematu bez tekstów i czasownika.	brak
GetTrStringCount	int	Zwraca liczbę obiektów w schemacie.	brak
GetTrDElementAtPos	WORD	Zwraca zakodowany element na pozycji Pos.	int Pos
GetTrDStringtAtPos	CString	Zwraca zakodowany string na pozycji Pos.	int Pos
GetTrDSemDescAtPos	int	Zwraca numer opisu semantycznego elementu Elem.	int Elem

4. Zakończenie

Zaimplementowana biblioteka funkcji dostępu umożliwia oprócz dostępu do zasobów bazy danych także analizę jej zawartości. Dzięki temu użytkownik może korzystać z gotowych rozwiązań i nie tracić czasu na analizę struktury bazy i sposobu zapisu danych w bazie. Dzięki implementacji w postaci kontrolki ActiveX narzędzie to jest dostępne w tych językach, które są przystosowane do korzystania z technologii AcitiveX. Przeprowadzono test mający na celu określenie szybkości działania funkcji odczytu danych i na komputerze klasy PC o częstotliwości pracy procesora 400 MHz otrzymano dla bazy Semsyn (baza obecnie zawiera dane o 9000 czasowników) czas odczytu danych dotyczących jednego czasownika około 0,3 s.

Opracowanie wykonano w ramach projektu badawczego 8 T11C 007 17 pt. „Translacja tekstów w języku polskim na język migowy", sposób wykorzystania danych został opisany w literaturze [4].

LITERATURA

1. Polański K.: Słownik syntaktyczno-generatywny czasowników polskich. Wyd. PAN, Warszawa 1980.
2. Grund D.: Komputerowa implementacja słownika syntaktyczno-generatywnego czasowników polskich. Studia Informatica, Gliwice 2000
3. Kuglinski D., Wingo S., Shepherd G.: Programming Microsoft Visual C++. Wyd. Microsoft Press, Redmond 1998.
4. Suszczańska N., Szmal P., Szczepanowski B.: Koncepcja tłumaczenia komputerowego z języka polskiego pisanego na język migowy. Postscsriptum, 1999, nr 27-29, ss. 63-72.

Recenzent: Dr hab. Zygmunt Vetulani Prof. UAM

Wpłynęło do Redakcji 28 listopada 2001 r.

Abstract

The article presents a software library implented as an ActiveX control. The library contains procedures and functions providing access to database of syntactic dictionary of Polish verbs. Section 1 presents fundmental informations on dictionary usages and the purpose of the described library. Section 2 describes the construction and concept of cooperation of modules, which the library consists of. The algorithm of *Semsyn* database access is depicted and illustrated with examples of its utilization in practice. A decoder of sentece schemes, which is a part of the library, is discussed and examples of scheme encoding are provided. Section 3 provides a detailed list of available library functions with descriptions of their operation. The article is ended with a summary and conclusions on the functioning of the library and methods of its application.