УДК 681.3.06

В.Г. Левицький, аспір.
Житомирський інженерно-технологічний інститут

ОПИС МОВИ КОРИСТУВАЧА В ПРОГРАМНОМУ СЕРЕДОВИЩІ ГЕНЕРАЦІЇ МОВНИХ ПРОЦЕСОРІВ

(Представлено к.т.н., доц. М.М. Колодницьким)

Розглянуто особливості організації роботи користувача в програмному середовищі генерації мовних процесорів "FancyCC 3.0". Запропоновано інтерфейс програмного середовища і синтаксис опису мови користувача. Коротко описані деталі реалізації автоматизованої побудови компілятора. Наведено ряд екранних форм для ілюстрації послідовності дій розробника трансляторів. Викладено також інформацію щодо сучасного становища в області автоматизованої генерації компіляторів.

Проблема автоматизації створення мовних процесорів виникла майже одночасно з початком широкого використання мов програмування та необхідністю створення для них трансляторів. Для того щоб застосувати до створення компіляторів більш системні методи в 50-х та 60-х роках до проблем, що пов'язані з формальними мовами та їх обробкою, почав широко застосовуватися математичний підхід. Із розвитком теоретичних побудов такої галузі системного програмування, як розробка трансляторів, почали виникати перші програмні засоби генерації мовних процесорів. Одним з найбільш поширених систем такого роду стали AT&T Lex та Yacc [13–16] – класичні генератори синтаксичних та лексичних аналізаторів, які входять у комплект поставки операційної системи Unix. Цей інструментальний засіб став, практично, стандартом, являючи собою перше вдале наближення до вирішення проблеми автоматизованої побудови трансляторів. Протягом багатьох років лише два шляхи лежали перед розробниками мовних процесорів. У тих бажаних та, зазвичай, недосяжних випадках, коли мова є порівняно простою, – слід використовувати Yacc. Якщо ж після обробки завдання цільового користувача розробник проблемно-орієнтованої мови (або мови програмування) приходив до висновку, що мова занадто складна, наприклад, потребує ускладнених процедур обробки помилок, а питання ефективності є критичним, все знову поверталося на десятиліття назад і синтаксичний аналізатор кодувався від початку до кінця. Слід, мабуть, погодитися, що таке інженерне рішення, яке справді було вдалим у 70-х роках, вже давно не задовольняє сучасний рівень розвитку інформаційних комп'ютерних технологій.

До теперішнього часу розробки середовищ генерації мовних процесорів продовжуються і останні вдалі розробки датуються вже дев'яностими роками. До найбільш відомих інструментальних засобів слід віднести пакети програм Cocktail, Gentle, PCCTS, COCOM, Eli, Amsterdam Compiler Kit, а також генератори лексичних і синтаксичних аналізаторів Anagram, Bison, Byacc, Coco, Cogencee, Depot4, Eag, Flex, Holub, Lisa, Llgen, Lalrgen, Mango, Muskox, Newyacc, Precc, Rdp, Scangen, Visualparse++, Yacc, Yacc++, Yooc, Zlex, Zyacc тощо. Досить великий арсенал подібних програм [1, 2, 13–26] значно розширив функціональні можливості AT&T Lex та Yacc. Тому ніша подібних інструментів заповнена дуже щільно, а сучасний засіб автоматизованої побудови компіляторів повинен відповідати високим стандартам. Успіх такого роду програмного забезпечення залежить від того, наскільки ефективно вирішуються проблеми інтерфейсу з користувачем та генерації трансляторів у порівнянні з існуючими інструментальними засобами, та від втілення нових ідей, які розширюють область застосування програми.

Аналіз функціональності всього вражаючого за своїм об'ємом арсеналу компіляторів компіляторів дозволяє стверджувати, що потреба у створенні нових інструментальних засобів автоматизованої побудови мовних процесорів все ще існує [1, 2]. Тому протягом останніх років на кафедрі програмного забезпечення Житомирського інженерно-технологічного інституту ведуться роботи в області розробки нових програмних засобів генерації мовних процесорів [2–6].

Актуальними проблемами, які, на наш погляд, не знайшли поки що свого адекватного вирішення, є: зручний та нескладний графічний інтерфейс користувача, простота мови опису проблемно-орієнтованої мови користувача, підтримка та оптимізація використання більш ніж

однієї проблемно-орієнтованої мови користувача. Розглянемо, яким чином вирішуються ці проблеми в програмному середовищі генерації мовних процесорів "FancyCC 3.0".

Для того щоб мати зручний та гнучкий інструмент опису мов, транслятори для яких повинні розроблятися, була створена внутрішня мова програмного середовища. Вона використовує синтаксично прості конструкції з мінімальним набором метасимволів та дозволяє за допомогою розширеної форми Бекуса-Наура (РБНФ) формулювати задачі генерації мовного процесора у звичному для даної предметної області вигляді. Приклад опису мови користувача наведено на рис. 1. Текст опису задачі генерації мовного процесора у загальному випадку складається з декількох блоків, кожен з яких визначає лексику (за допомогою регулярних виразів) або синтаксис (за допомогою РБНФ) однієї з мов користувача. На початку блоків можна використовувати конструкцію "<ключове слово> <ідентифікатор>" для іменування мов та окремих лексичних та синтаксичних розділів. Тут ключовими словами є: **language**, **token**, **production**. Більш детальний виклад синтаксису мови опису формальних граматик можна знайти в [1, 2].

```
/* Sample grammar */

// SSçûê
language first;

// Ëâêñàíû
   token first;

     letter = [a-zA-Z_];
     digit = [0-9];
     identifier = letter ( letter | digit )*;
     sample = a{b}[c](~[d]|s);

// Ïðàâèëà ïîäñòàíîâêè
   production first;

     EXPR     -> TERM { '+' TERM };
     TERM     -> identifier = identifier;
```

Рис. 1. Приклад опису мови користувача на внутрішній мові програмного середовища

Однією з найбільш суттєвих особливостей внутрішньої мови програмного середовища "FancyCC 3.0" є мінімальна підтримка семантичних аспектів розробки транслятора. Така специфіка опису мови користувача обумовлюється авторською позицією щодо прогресу в такій галузі комп'ютерних наук, як опис семантики формальних мов. На наш погляд, найбільш вдалою формою опису семантики залишається мова програмування. Інші засоби та форми визначення семантики, що використовується в багатьох існуючих програмних засобах компіляторів компіляторів, є дуже незручними для використання та ускладнюють роботу розробника трансляторів замість того, щоб спрощувати її. Інша думка, яка вплинула на розробку програмного середовища "FancyCC 3.0", є та, що опис формальної мови користувача повинен проводитись в межах одної проблемно-орієнтованої мови, без використання фрагментів цільової мови програмування розробника трансляторів, які засмічують текст опису деталями реалізації.

Програмне середовище "FancyCC 3.0" використовує адаптивну процедуру [7–10] обробки множини споріднених проблемно-орієнтованих мов, що дозволило ефективно використати даний інструментальний засіб генерації компіляторів під час розробки підсистеми лінгвістичного забезпечення програмної системи "Dsr Open Lab 1.0" [11, 12].

Далі наведено екранні форми (рис. 2–10), які повністю ілюструють процес роботи розробника трансляторів у програмному середовищі "FancyCC 3.0". Послідовність прикладів у цілому відповідає послідовності дій, необхідних для розробки компілятора мови користувача. Спочатку користувач вибирає тип проекту опису мови або множини мов, визначаючи свою кінцеву мету, – розробка транслятора чи документування вже існуючої мови (рис. 3). Потім він описує проблемно-орієнтовану мову (рис. 4) та проглядає результати генерації компілятора (рис. 5–7) або документування (рис. 8). Архітектура програмного середовища та система контекстної допомоги наведені на рис. 10 та 11.

Рис. 2. Початок роботи з програмним середовищем генерації мовних процесорів

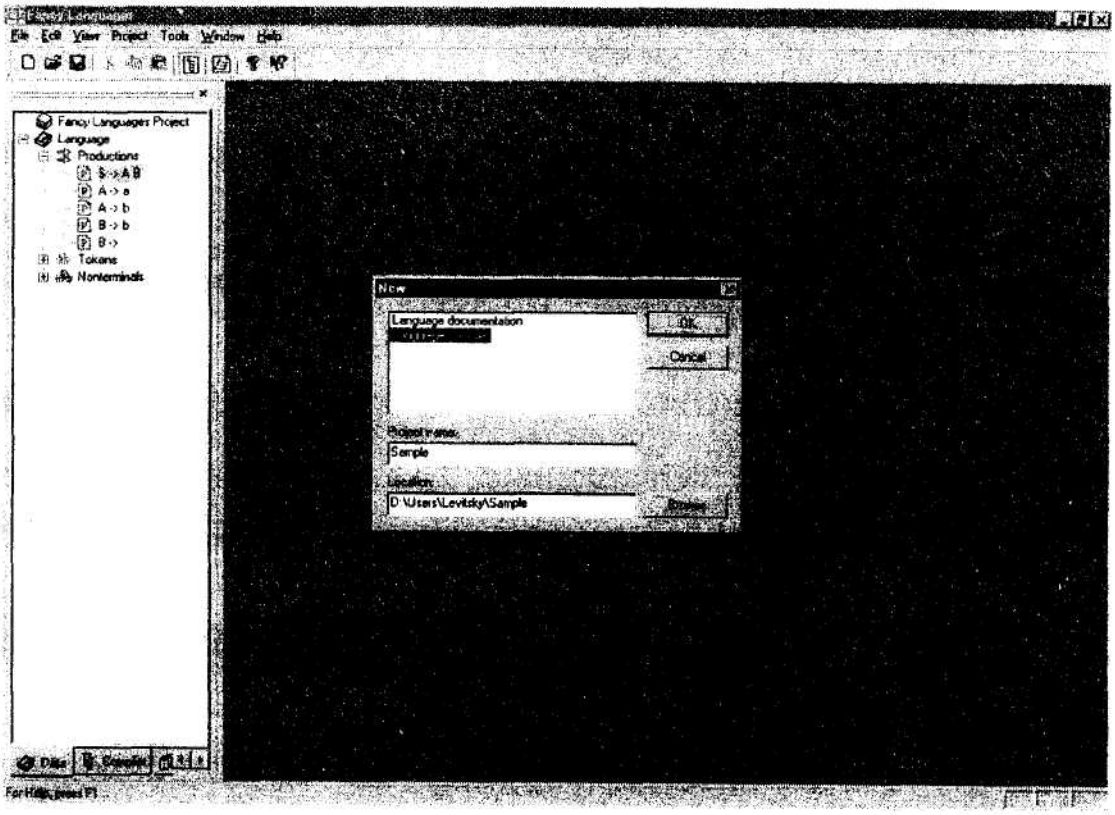


Рис. 3. Створення нового проекту опису мов користувача

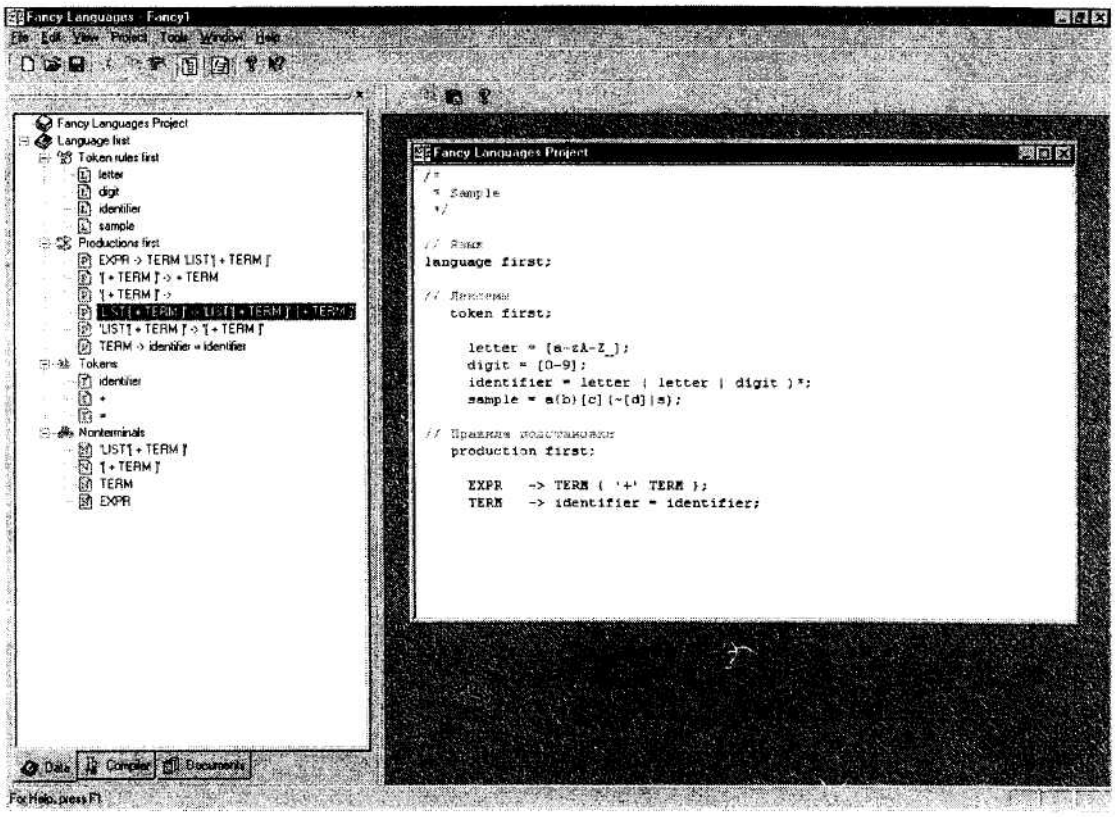
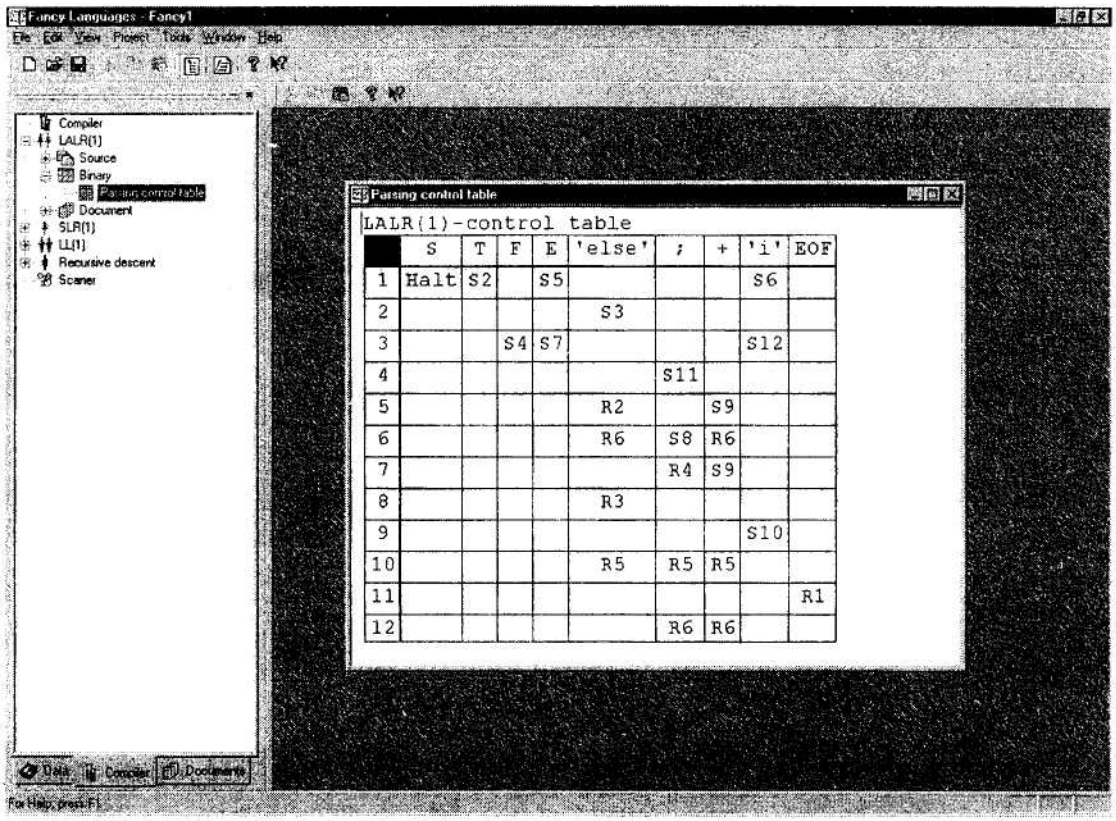


Рис. 4. Редагування опису мов користувача



	S	T	F	E	'else'	;	+	'i'	EOF
1	Halt	S2		S5				S6	
2					S3				
3			S4	S7				S12	
4						S11			
5					R2		S9		
6					R6	S8	R6		
7						R4	S9		
8					R3				
9								S10	
10					R5	R5	R5		
11									R1
12						R6	R6		

Рис. 5. Побудова LALR(1)-парсера

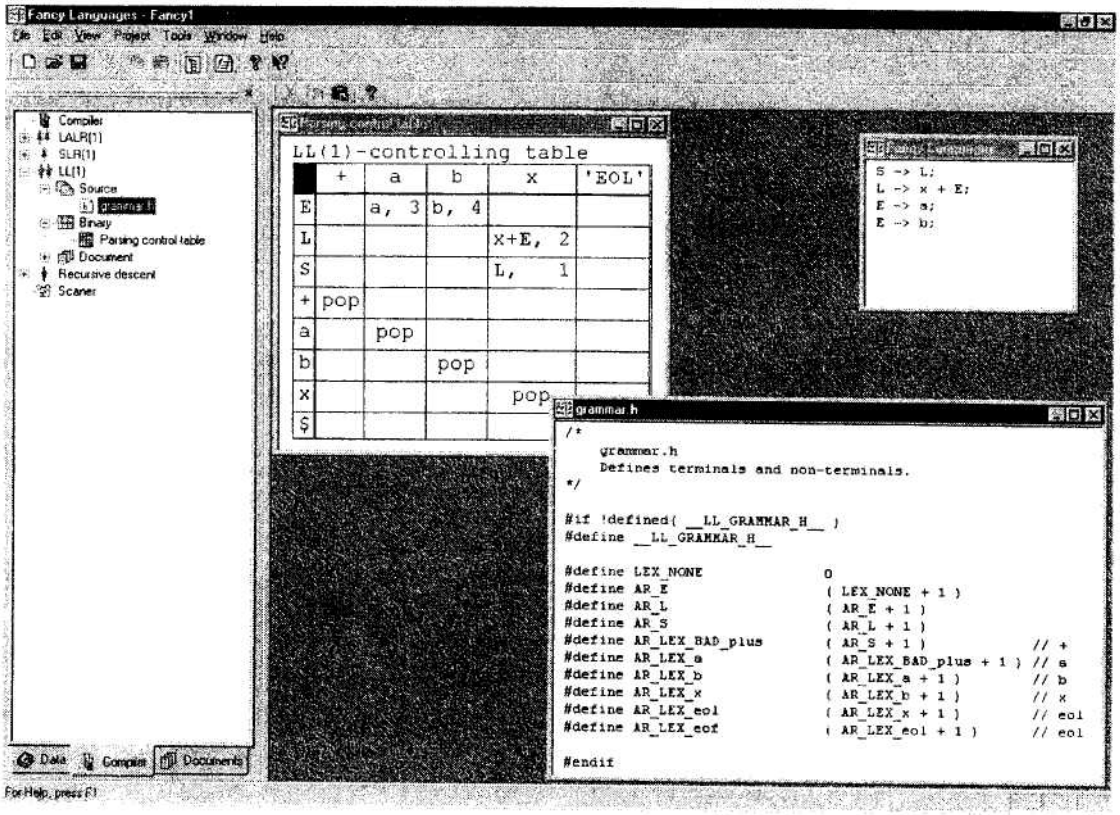


Рис. 6. Побудова LL(1)-парсера

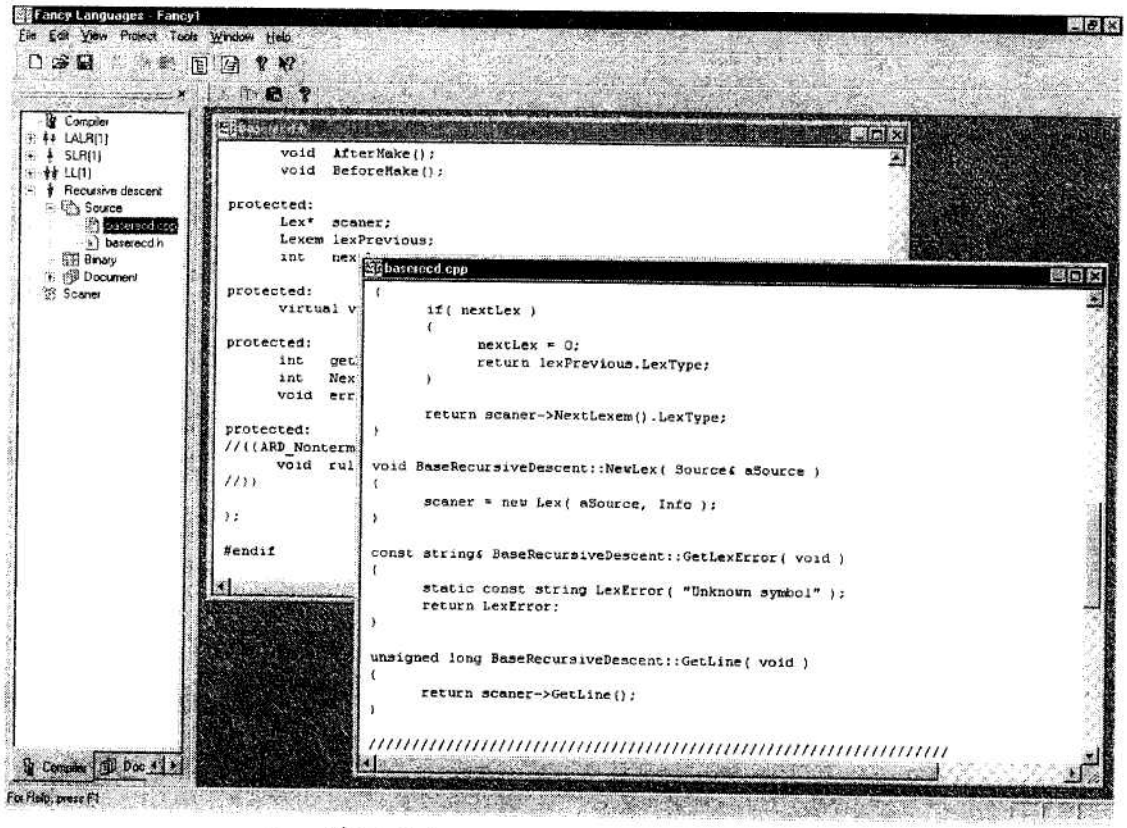


Рис. 7. Побудова парсера рекурсивного спуску

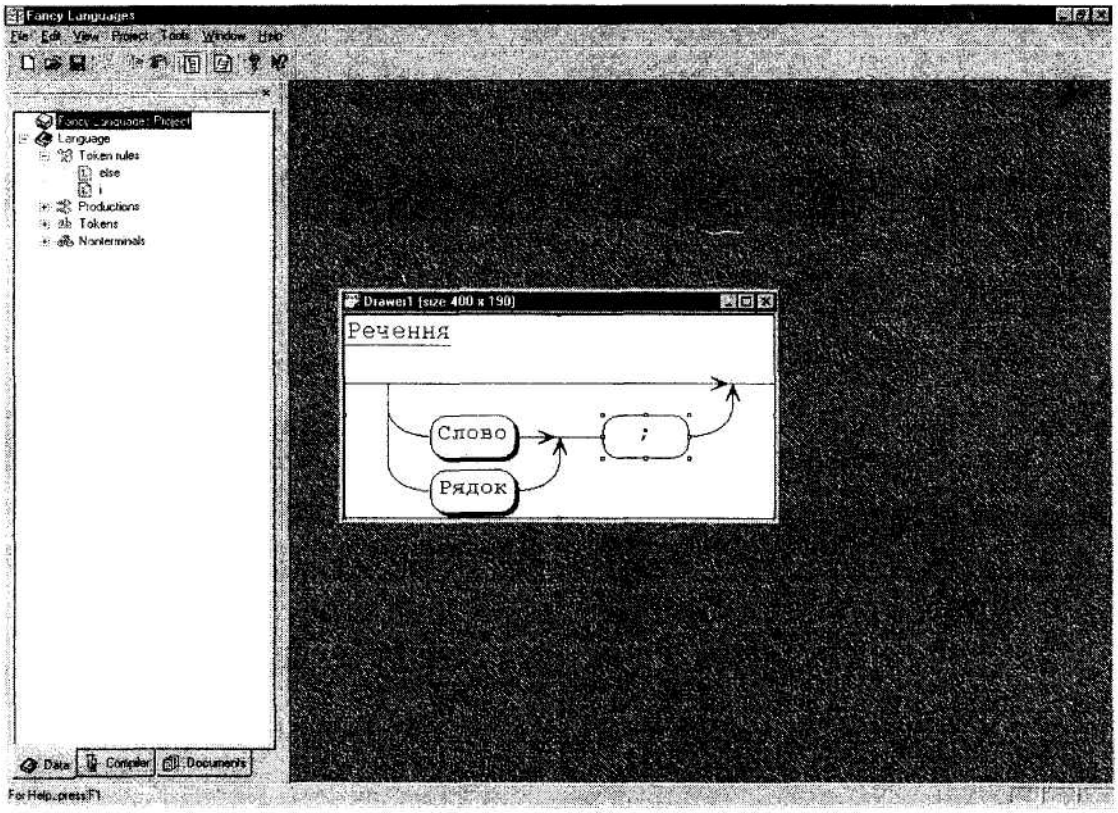
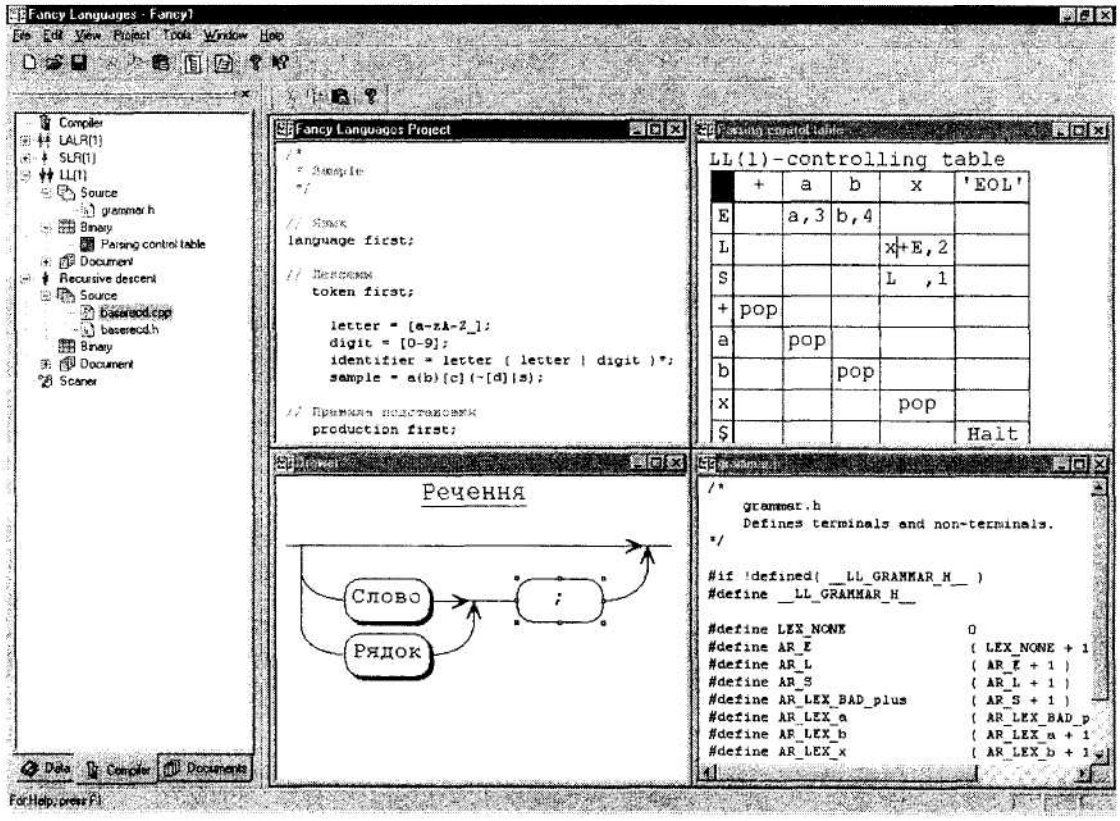


Рис. 8. Документування мови користувача



	+	a	b	x	'EOL'
E		a,3	b,4		
L				x+E,2	
S				L ,1	
+	pop				
a		pop			
b			pop		
x				pop	
$					Halt

Рис. 9. Відображення даних за допомогою OLE-серверів

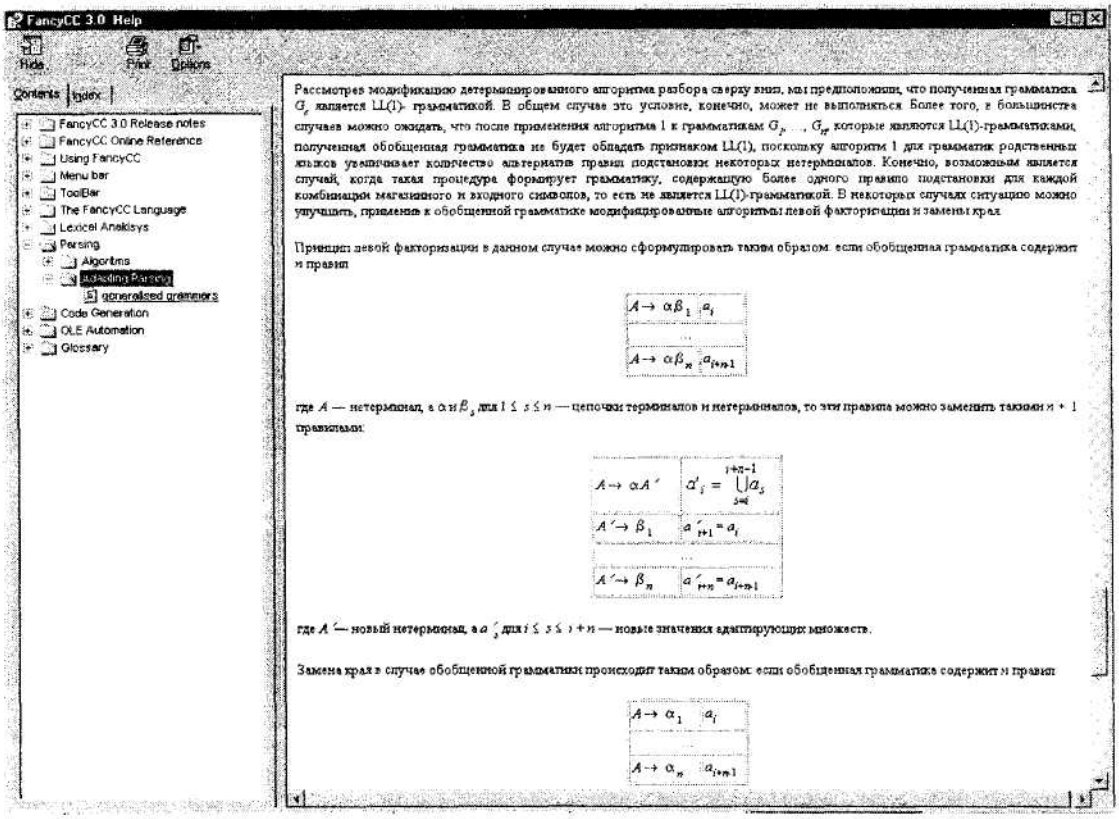


Рис. 10. Контекстна допомога

ЛІТЕРАТУРА:

1. *Колодницький М.М., Левицький В.Г.* Сучасні інструментальні засоби автоматизованої побудови мовних процесорів // Проблемы программирования, спец. выпуск. – № 1–2. – Материалы 2-й Международной научн.-практ. конф. по программированию "УкрПрог'2000", 23–26 травня 2000 р. – Київ, 2000. – С. 345–350.
2. *Левицький В.Г.* Мова визначення вхідних даних програмного середовища генерації мовних процесорів "FANCYCC 3.0" // Вісник ЖІТІ. – 2000. – № 12. – С. 226–233.
3. http://scilab.ziet.zhitomir.ua/comp/ling/ling_all.php.
4. *Колодницький М.М., Левицький В.Г., Рожик О.А.* Автоматизація побудови компілятора мови опису математичних моделей динамічних систем // Вісник ЖІТІ. – 1996. – № 4. – С. 138–152.
5. *Левицький В.Г.* Автоматизація процесу конструювання трансляторів за допомогою інструментального засобу "FANCYCC 2.0" // Праці 2-ї національної науково-практичної конференції студентів та аспірантів "Системний аналіз та інформаційні технології", 28–30 червня 2000 р. – Київ, 2000. – С. 164–167.
6. *Левицький В.Г.* Спосіб автоматичної генерації детермінованих скінченних автоматів за текстовим описом без попередньої обробки регулярних виразів // Вісник ЖІТІ. – 2000. – № 13. – С. 192–196.
7. *Колодницький М.М., Левицький В.Г.* Адаптивна організація лінгвістичного забезпечення програмного комплексу "DSR Open Lab 1.0" // Праці 1-ї Міжн. наук.-практ. конф. з програмування "УкрПрог'98", 2–4 вересня 1998 р. – Київ, 1998. – С. 145–155.
8. *Левицький В.Г.* Одна реалізація синтаксичного аналізу текстів споріднених мов за допомогою модифікованого алгоритму SLR(1)-розбору // Вісник ЖІТІ. – 1999. – № 9. – С. 225–230.
9. *Левицький В.Г.* Застосування формалізму породжуючих граматик до аналізу споріднених мов математичного моделювання складних систем // Праці 1-ї

національної науково-практичної конференції студентів та аспірантів "Системний аналіз та інформаційні технології", 28–29 червня 1999 р. – Київ, 1999. – С. 32–33.

10. *Колодницкий Н.М., Левицкий В.Г.* Применение формализма порождающих грамматик к анализу родственных языков // Кибернетика и системный анализ. – 1999. (в друці).
11. *Колодницький М.М., Левицький В.Г.* Типологія архітектури інтерфейсу користувача прикладної програмної системи "DSR Open Lab 1.0". Частина II. Лінгвістичне забезпечення // Проблемы программирования. – 1999. – Вып. 3–4. (в друці).
12. *Kolodnytsky M., Kovalchuk A., Kuryata S., Levitsky V.* The Mathematical Software Implementation for Computational Algebra and Number Theory // Proceedings of the 4th Asian Symposium on Computer Mathematics, December 17–21, Chiang Mai, Thailand, 2000. – (in press)
13. *Кристиан К.* Введение в ОС UNIX. – М.: Финансы и статистика, 1985. – 320 с.
14. *Тихомиров В.П., Давыдов М.И.* ОС ДЕМОС: инструментальные средства программирования. – М.: Финансы и статистика, 1988. – 204 с.
15. *Johnson S.C.* YACC – yet another compiler-compiler. Comp. Sci. tech. rep. № 32. Bell Laboratries: Murray Hill, New Jersey, 1975.
16. *Lesk M.E.* Lex – lexical analyzer generator. Comp. Sci. tech. rep. № 39. Bell Laboratries: Murray Hill, New Jersey, 1975.
17. *Grosch J., Emmelmann H.* A Tool Box for Compiler Construction, Technical Report No. 20, Gesellschaft fьr Mathematik und Datenverarbeitung mbH, Forschungsstelle an der Universitдt Karlsruhe, 1990.
18. *Schrцer F.W.* The GENTLE Compiler Construction System, R. Oldenbourg Verlag, Munich and Vienna, 1997. – 142 p.
19. *Parr T.J.* Language Translation Using PCCTS and C++. A Reference Guide. Automata Publishing Company, 1993. – 310 p.
20. *Tanenbaum A.S.* et al. Amsterdam Compiler Kit documentation, Rapport nr IR-90, Vrije Universiteit, Amsterdam, 1984.
21. *Sonnenschein M.* Graph translation schemes to generate compiler parts, ACM Trans. Program. Lang. Syst. 9, 4 (Oct. 1987). – Pp. 473–490.
22. *Pleban U.F. and Lee P.* An automatically generated, realistic compiler for imperative programming language, Proceedings of the SIGPLAN'88 conference on Programming Language design and Implementation, 1988. – Pp. 222 –232.
23. *Andrews K., Henry R.R. and Yamamoto W.K.* Design and implementation of the UW Illustrated compiler, Proceedings of the SIGPLAN'88 conference on Programming Language design and Implementation, 1988. – Pp. 105–114.
24. *Henry R.R., Whaley K.M. and Forstall B.* The University of Washington illustrating compiler, Proceedings of the conference on Programming language design and implementation, 1990. – Pp. 223–233.
25. *Sloane A.M.* An evaluation of an automatically generated compiler, ACM Trans. Program. Lang. Syst. 17, 5 (Sep. 1995). – Pp. 691–703.
26. *Jourdan M., Parigot D., Julie C., Durin O. and Le Bellec C.* Design, implementation and evaluation of the FNC-2 attribute grammar system, Proceedings of the conference on Programming language design and implementation, 1990. – Pp. 209–222.

ЛЕВИЦЬКИЙ В'ячеслав Георгійович – аспірант Житомирського інженерно-технологічного інституту.

Наукові інтереси:

- комп'ютерні інформаційні технології;
- моделювання і розв'язок задач за допомогою обчислювальної техніки;
- використання обчислювальної техніки в навчальному процесі;
- побудова компіляторів.

Подано 25.10.2000