ОПРЕДЕЛЯНЕ НА ОБЕКТИ, ОЗНАЧАВАНИ С ИДЕНТИФИКАТОРИ ПРИ ПРЕВОД НА ПРОГРАМИ НА ФОРТРАН ЧРЕЗ ТРАНСЛАТОРА ФОР32

Маргарита Бърнева

I. ПОСТАНОВКА НА ПРОБЛЕМАТА

Една от основните задачи на всеки транслатор от ФОРТРАН е определянето на вида на обектите, означавани с идентификатори. За решаването на тази задача е необходимо да се фиксират видовете допустими обекти и техните характеристики и да се построи алгоритъм за разпознаване на вида на всеки срещан в програмата обект. При това трябва да се има пред вид, че идентификаторът на даден обект обикновено се среща в програмата многократно и че са възможни противоречиви употреби на идентификатори, дължащи се на грешки в програмата.

Класификацията на обектите по видове не може да бъде произволна. Тя се определя от езиковите категории и от изискванията на транслационните схеми.

Желателно е класификацията да отразява колкото може по-пълно свойствата на употребените в програмата обекти, но същевременно да не утежнява съществено схемите на транслация.

Алгоритъмът за разпознаване и класифициране на обектите определя в значителна степен качеството на транслатора от гледна точка на индикациите на грешки.

В настоящата работа се излага класификацията на различните обекти и алгоритъмът за разпознаването им, реализиран в транслатора ФОР32 [2].

Входният език на транслатора ФОР32 се основава на BASIC FORTRAN IV и е описан подробно в [1].

Една програма на входния език представлява съвкупност от главна програмна част (ROUTINE) и допълнителни програмни части — подпрограми (SOUBROUTINE) и функции (FUNCTION). Всяка програмна част се транслира самостоятелно и независимо от другите части. От последното обстоятелство произтичат следните две следствия:

а) налагат се някои ограничения на езика (например недопускане използуването на функции без параметри и др.);

б) налага се отчитане на допълнителни характеристики за обектите, които позволяват да се контролират програмите при тяхното взаимодействие.

В най-общия случай една програмна част се състои от следните видове оператори:

заглавен оператор;
описателни оператори;
аритметични функции;
изпълняеми оператори;
краен оператор.

С изключение на някои оператори (END, STOP, RETURN и др.) всички оператори или указват определени действия върху обекти, или описват някои свойства на обекти. Възможно е да се укаже действие едновременно с описание на свойства на обекти. Например операторът CALL A(x) освен съответното действие определя, че идентификаторът A означава име на подпрограма. Поради това при определянето на вида на даден обект се вземат пред вид както описателните, така и изпълняемите оператори.

II. КЛАСИФИКАЦИЯ НА ОБЕКТИТЕ, ОЗНАЧАВАНИ С ИДЕНТИФИКАТОРИ

Всички допустими обекти[1] се отнасят към четири класа:

процедури;
формални параметри;
масиви;
прости променливи).

С изключение на подпрограмите и главните програмни части обектите се характеризират допълнително с тип, отнасящ се до начина на представяне на стойностите, които те могат да получат. Допускат се обекти от цял тип и от реален тип.

По-долу се описват видовете обекти във всяка класа и се излагат някои съображения, свързани с възприетата класификация. Приведени са също така кодовете на обектите. Тези кодове се употребяват в следващия параграф при описване на алгоритъма за разпознаване.

В табл. 1а и 1б са дадени възможните видове обекти от класата на процедурите.

Таблица 1а

Род на процедурата		Код на процедура, изпо зувана		
		само в EXTERNAL	само явно	и в EXTERNAL, и явно
Функция	от цял тип	101	104	107
	от реален тип	102	105	108
Подпрограма		103	106	109

[1] Тук и по-долу обект, означаван с идентификатор, се нарича накратко само обект.

Таблица 16

Род на процедурата			Код на процедурата, когато	
			на идентификатора й не се присвоява стойност	на идентификатора й се присвоява стойност
Главна програмна част			110	—
Допълнителна програмна част — подпрограма			111	—
Допълнителна програмна част—функция	от цял тип		112	114
	от реален тип		113	115
Вградена функция			116	—
аритметична функция	използва друга аритметична функция	цял тип	117	—
		реален тип	118	—
	не използва друга аритметична функция	цял тип	119	—
		реален тип	120	—

Към класата на процедурите спадат:

главна програмна част;

подпрограма SUBROUTINE;

подпрограма FUNCTION;

вградени функции;

аритметични функции;

подпрограми и функции, които се използват в дадената програмна част.

Класификацията на процедурите се основава на следните допълнителни признаци:

тип на процедурата;

включена ли е процедурата в оператор CALL;

присвоява ли се стойност на името на процедурата.

Разграничаването на аритметичните функции, при които в определящия ги израз участвува вече описана артитметична функция, се налага от схемите на транслация. При транслация на различни аритметични функци за работни полета се определят едни и същи клетки с цел да се използва по-оптимално паметта. Този принцип се нарушава само в случай, когато в израза се използват други аритметични функции.

За всеки вид процедура се отчита освен това броят на нейните параметри. Тази характеристика не е отразена в таблиците.

Вградените функции се считат предварително фиксирани и определени по тип и брой на параметрите.

Видовете формални параметри и техните кодове се дават в табл. 2а и 2б. В табл. 2б явно ползуване на формален параметър означава

Таблица 2а

Тип на формалния параметър	Код на формален параметър			
	променлива			маенв (заместван по име)
	замествана по стойност	замествана по име		
		включена в LIST	невключена в LIST	
Цял	201	204	207	210
Реален	202	205	208	211
С неявно определен тип	203	206	209	212

Таблица 2б

Тип на формалния параметър		Код на формален параметър, използуван		
		само в EXTERNAL	само явно	и в EXTERNAL, и явно
Функция	цял тип	213	216	219
	реален тип	214	217	220
Подпрограма		215	218	221

ползуване чрез указател (при функции) или чрез оператор CALL (при подпрограми).

Кодът 215 означава произволна програмна част (подпрограма или функция), която се ползува само в EXTERNAL, т. е. кодът 215 включва като частни случаи кодовете 213 и 214.

Временно некласифициран формален параметър се означава с код 200 и не е включен в таблиците.

При анализа на формалните параметри се взема под внимание:

какви фактически параметри могат да им съответствуват — променливи, масиви, функции, подпрограми;

типът на параметрите;

включени ли са в оператор EXTERNAL;

включени ли са в оператор LIST;

начинът на замяна с фактическите параметри (по име или по стойност).

Последната характеристика позволява да се извършва допълнителен контрол при взаимодействие между програмни части и в случай, че на формален параметър, заместван по име, се съпоставя константа или израз, да се индицира грешка.

Включеният в езика оператор LIST е предназначен да облекчава проверката на програмите. В този оператор се задава списък от прости променливи. При всяко изменение на стойността на коя да е от простите променливи от списъка се отпечатват идентификаторът и стойността на променливата. За това е необходимо идентификаторите на тези променливи да бъдат запазени в транслираната програма. Това налага да бъдат различавани променливи, които са включени в оператор LIST.

Към класата на формалните параметри се отнасят и формалните параметри на аритметичните функции. Предполага се, че те са само прости променливи с неявно определен тип. Код на този вид формални параметри не е посочен, тъй като тяхната обработка не е включена при изложението на алгоритъма за разпознаване.

В табл. 3 са посочени различните масиви. При класификацията на масивите и на простите променливи се взима под внимание участието на

Таблица 3

Тип на масива		Код на масив	
		невключен в COMMON област	включен в COMMON област
Явно определен	цял	301	304
	реален	302	305
Неявно определен		303	306

съответните идентификатори в списъците на оператори COMMON и EQUIVALENCE, тъй като тези характеристики влияят при разпределението на оперативната памет.

Всеки обект от класата на масивите има още две важни характеристики, които не са отразени в таблицата. Това са размерността и горните граници на индексите.

В табл. 4 са дадени допустимите видове прости променливи. За признаци, по които става класифицирането на една проста променлива, са взети:

тип на променливата;

включена ли е променливата в оператор LIST;

включена ли е променливата в оператор COMMON и EQUIVALENCE,

Таблица 4

Тип на простата променлива		Код на променливата				
		невключена в COMMON област				включена в COMMON област
		невключена в оператор LIST		включена в оператор LIST		
		невключена в EQUIVALENCE	включена в EQUIVALENCE	невключена в EQUIVALENCE	включена в EQUIVALENCE	
Явно определен тип	цял	401	404	407	410	413
	реален	402	405	408	411	414
Неявно определен тип		403	406	409	412	415

III. АЛГОРИТЪМ ЗА РАЗПОЗНАВАНЕ ВИДА НА ОБЕКТИТЕ

Определянето на вида на обектите се извършва по време на първия преглед (в права посока) на превежданата програма. Създаденият алгоритъм протича в два етапа. Всеки етап се управлява от предварително подготвена таблица. Първият етап се отнася за обекти в описателните оператори, а вторият — за обекти в изпълняемите оператори.

Основно положение в алгоритъма е приемането, че първата поява на обекта определя напълно или до известна степен вида му. Всяка следваща поява на обекта в същия или друг оператор евентуално допринася за уточняване на вида му или позволява да се констатира неправилна употреба на обекта.

Алгоритъмът предполага, че:

различните обекти се означават с различни идентификатори;

могат да се използват идентификатори с неограничен брой символи, но идентификаторите, на които първите пет символа съвпадат, се считат за еднакви;

идентификаторите, използувани за означаване на вградените функции, не могат да се използват за означаване на други обекти;

идентификатори, започващи с резервираните думи на езика, не се използват.

В първия етап на алгоритъма за код на всеки обект се пресмятат последователни стойности y_n, при всяка поредна поява $n=1, 2, 3, \ldots$ на обекта, по формулата

$$y_n = Y(y_{n-1}, \omega). \tag{1}$$

Таблица 5

q_{n-1} \ ω	INTEGER	REAL	LIST	EQUIVALENCE	COMMON	DIMEN-SION	EXTERNAL
0	401	402	409	406	415	303	103
101	—	—	—	—	—	—	—
102	—	—	—	—	—	—	—
103	101	102	—	—	—	—	—
200	201	202	206	—	—	212	215
201	—	—	204	—	—	210	213
202	—	—	205	—	—	211	214
203	—	—	—	—	—	—	—
204	—	—	—	—	—	—	—
205	—	—	—	—	—	—	—
206	204	205	—	—	—	—	—
207	—	—	—	—	—	—	—
208	—	—	—	—	—	—	—
209	—	—	—	—	—	—	—
210	—	—	—	—	—	—	—
211	—	—	—	—	—	—	—
212	210	211	—	—	—	—	—
213	—	—	—	—	—	—	—
214	—	—	—	—		—	—
215	213	214	—	—	—	—	—
216	—	—	—	—	—	—	—
217	—	—	—	—	—	—	—
218	—	—	—	—	—	—	—
219	—	—	—	—	—	—	—
220	—	—	—	—	—	—	—
221	—	—	—	—	—	—	—
301	—	—	—	301	304	—	—
302	—	—	—	302	305	—	—
303	301	302	—	303	306	—	—
304	—	—	—	304	—	—	—
305	—	—	—	305	—	—	—
306	304	305	—	—	—	—	—
401	—	—	407	404	413	301	101
402	—	—	408	405	414	302	102
404	—	—	410	404	413	—	—
405	—	—	411	405	414	—	—
406	404	405	412	406	415	—	—
407	—	—	—	410	—	—	—
408	—	—	—	411	—	—	—
409	407	408	—	412	—	—	—
410	—	—	—	410	—	—	—
411	—	—	—	411	—	—	—
412	410	411	—	412	—	—	—
413	—	—	—	413	—	304	—
414	—	—	—	414	—	305	—
415	413	414	—	415	—	306	—

Описателният оператор, в който се среща за n-ти път обектът, е означен с ω. Приема се, че $y_0=0$.

Двуаргументната функция Y е зададена таблично в табл. 5. Видовете описателни оператори са указани в таблицата.

На втория етап на алгоритъма продължава пресмятането на последователни кодове за всеки обект, срещан в първия етап, както и за новопоявили се обекти. Кодовете са пресмятат по формулата

$$y_n=Z(y_{n-1}, b), \tag{2}$$

като за новопоявилите се обекти (както в (1) се приема, че $y_0=0$).

В този етап при всяка поредна поява на изследвания обект се пресмятат три функции p_1, p_2, p_3. Всяка от тях взема стойност 0 или 1, в зависимост от отношението на изследвания обект с определени други езикови елементи от оператори, в които обектът се появява. По-точно функцията p_1 приема стойност 1, когато на обекта се присвоява стойност; p_2 приема стойност 1, когато обектът се използва чрез оператор CALL; p_3 е 1, когато обектът се използва като указател на функция или име на масив.

Аргументът b в (2) се пресмята по формулата

$$b=4p_1+2p_2+p_3;$$

от осемте възможни стойности на b, съответствуващи на различните комбинации на признаците p_1, p_2, p_3, са сместени само пет: $b=0$, 1, 3, 4, 5.

Функцията Z е зададена чрез табл. 6. В случаите, когато при дадени q_{n-1} и b са посочени по два кода, за q_n се избира първият, когато обектът е от цял тип, и вторият, когато обектът е от реален тип.

Таблица 6

q_{n-1} \ b	0	1	3	4	5
101	101	107	—	—	—
102	102	108	—	—	—
103	103	107/108	109	—	—
104	—	104	—	—	—
105	—	105	—	—	—
106	—	—	106	—	—
107	107	107	—	—	—
108	108	108	—	—	—
109	109	—	109	—	—
110	—	—	—	—	—
111	—	—	—	—	—
112	—	112	—	114	—
113	—	113	—	115	—
114	—	—	—	114	—
115	—	—	—	115	—
116	—	116	—	—	—
117	—	117	—	—	—
118	—	118	—	—	—

Продължение на таблица 6

q_{n-1} \ b	0	1	3	4	5
119	—	119	—	—	—
120	—	120	—	—	—
200	203	216/217	218	209	—
201	201	216	—	207	—
202	202	217	—	208	—
203	203	—	—	—	—
204	204	—	—	—	—
205	205	—	—	205	—
206	206	—	—	206	—
207	207	—	—	207	—
208	208	—	—	208	—
209	209	—	—	209	—
210	*	210	—	**	210
211	*	211	—	**	211
212	*	212		**	212
213	213	219	—	—	—
214	214	214	—	—	—
215	215	219/220	221	—	—
216	—	216	—	—	—
217	—	217	—	—	—
218	—	—	218	—	—
219	219	219	221	—	—
220	220	220	—	—	—
221	—	—	—	—	—
301	*	301	—	**	301
302	*	302	—	**	302
303	*	303	—	**	303
304	*	304	—	**	304
305	*	305	—	**	305
306	*	306	—	**	306
401	401	—	—	401	—
402	402	—	—	402	—
403	403	—	—	403	—
404	404	—	—	404	—
405	405	—	—	405	—
406	406	—	—	406	—
407	407	—	—	407	—
408	408	—	—	408	—
409	409	—	—	409	—
410	410	—	—	410	—
411	411	—	—	411	—
412	412	—	—	412	—
413	413	—	—	413	—
414	414	—	—	414	—
415	415	—	—	415	—
0	403	104/105	106	403	—

Случаите, когато в табл. 5 и табл. 6 вместо съответни кодове са поставени тирета, съответствуват на неправилна употреба на обекта. При тези случаи се записва старият код на обекта, т. е. полага се $q_n = q_{n-1}$, като на мястото на обекта в транслирания оператор се отбелязват кон-

статираните грешки. При следващата фаза на транслатора тази информация се транслира в индикатори за грешки.

От таблиците се вижда, че могат да бъдат констатирани около 400 различни грешки. Такова разнообразие в същност би довело до разход на оперативна памет, без да подпомага съществено програмиста. Затова грешките са групирани и се съобщават около 20 различни индикации.

Групирането на грешките по един или друг начин се извършва, като се въведе код за всяка група грешки и този код се употребява вместо съответните на грешките тирета в табл. 5 и 6.

В табл. 6 символът * означава, че $q_n=q_{n-1}$, когато обработваният оператор е оператор WRITE. В противен случай символът * е равносилен на символа тире (—). Аналогично символът * * означава, че $q_n=q_{n-1}$, ако се обработва обект от оператор READ, и грешка в противен случай.

В таблиците не са отразени проверки на изисквания, наложени от по-частни характеристики, като например тип, размерност на масивите и др.

В резултат от работата на програмните блокове, реализиращи тези алгоритми, за правилно употребените обекти се съставят таблици, съдържащи необходимата информация за работата на втората фаза от транслатора на преведената програма.

ЛИТЕРАТУРА

1. ФОР32: Транслатор от ФОРТРАН за машината „МИНСК-32". Ръководство за програмиста. С., 1973. БАН, ИММ с ИЦ (вътрешно издание).
2. Бърнев, П., Бърнева, М., Добрев, Д. Д., Томов, В.: Транслятор ФОР32 с языка ФОРТРАН для ЭВМ „Минск-32". Сердика (под печат).

Постъпила на 14. XII. 1974 г.

DETERMINING OF ELEMENTS, DENOTED BY IDENTIFIERS WHEN TRANSLATING FORTRAN PROGRAMS BY THE FOR32 TRANSLATOR

M. Bârneva

(SUMMARY)

The paper treats the problem of determining the type of program elements in FORTRAN programs, denoted by identifiers. A classification of these elements is performed (tables 1 — 4) and some problems, concerning their recognition, are discussed.

The corresponding recognition algorithms, controlled by previously prepared tables (5, 6) with two entry points, which are realized in the FOR32 translator from FORTRAN for the MINSK 32 computer, are described.