АНАЛИЗ НА РЕШЕНИЕТО НА ЗАДАЧА VILLIE

На практика почти цялата задача се свежда до обработка на числови изрази. Естествено, целта ни е да е ги приведем в удобен вид, какъвто е обратният полски запис. Това обаче е достатъчно само за първите 20% от тестовете.

Ако няма рекурсивни повиквания във функциите, т.е. една функция не използва себе си и няма функции, които да зависят и двете една от друга, тогава техните изрази могат да се сведат до такива, които не съдържат повиквания на други функции. Остава единствено буквите да се заменят със стойността на аргументите. Едно такова решение би получило 40 точки.

Ако не всяка функция може да се сведе до краен израз, тогава е възможно решението да се напише рекурсивно – при срещане на функция в даден израз, да се изчисли стойността рекурсивно. Стига да не е прекалено дълбока рекурсията, този метод трябва да се справи със задачата. Такова решение би получило 60 точки. Забележка: Дори рекурсията да не е дълбока, броят на повикванията на функции може да стане много голям.

Какво става ако рекурсията стане прекалено дълбока? Тогава ни трябва итеративен начин на изчисление. Това може да бъде постигнато по следният начин:

- Всички изрази се превръщат в обратен полски запис.
- Дефинират се оператори за начало, край и повикване на функция.
- При повикване на функция, изчислението се „премества“ в друга част на израза.

Това което трябва да се получи е нещо като друга програма, на език от по-ниско ниво. Резултатът накрая се изчислява почти по обикновения алгоритъм за обратен полски запис. Примерна реализация:

Оператор	Действие
`+`	Взема най-горните 2 числа от резултатния стек, събира ги, и връща в стека резултата.
`-`	Подобно на +. Първото число на върха на стека е това, което изваждаме.
`*`	Подобно на +.
`/`	Подобно на –.
`<`	Взима най-горните 2 числа от резултатния стек и връща в стека 1, ако второто е по-малко от първото, или 0 в противен случай.
`>`	Подобно на <.
`=`	Подобно на <.
`<=`	Подобно на <.
`>=`	Подобно на <.
`func a n`	Начало на функция. След това има 2 аргумента – брой аргументи на функцията и брой операции. При срещане, прескача функцията.
`jump n`	Прескача n на брой операции

`jump_zero n`	Взима числото на върха на стека. Ако е 0, прескача n на брой операции.
`call p`	Повиква функция с начало позиция p. Аргументите трябва предварително да са сложени в стека, като първият аргумент е числото на върха на стека.
`return`	Изтрива аргументите на текущата функция и оставя само резултата.
`con n`	Въвежда константата n в стека.
`arg n`	Въвежда n-я аргумент в стека.
`stop`	Край на програмата.

Така за да повикаме функция, въвеждаме в стека всичките й аргументи, в обратен ред. Най-удобно е ако имаме други стекове, в които на върха се помни къде е краят на аргументите, къде в програмата трябва да се върнем след края на функцията и колко са на брой аргументите, които трябва да се изтрият.

if изразите се обработват по следния начин: Изчислява се условието, използва се jump_zero, за да се прескочи изчислението на първия израз, ако резултатът е 0. В края на първия израз се прескача втория чрез jump.

Всеки израз в скоби е най-добре да се обработи рекурсивно, а не с обикновения алгоритъм за обратен полски запис, поради възможността да има if вътре.

Тъй като дадената програма не е особено голяма, можем да си „позволим“ рекурсия и по-голяма сложност при обработването й. Изпълнението на получения „израз“ след това остава тривиално. Обработка на примерна програма:

```
fun fact(n)=if n<=1 then 1 else n*fact(n-1);
fact(10);
```

Позиция	Операция
0	`func`
1	`1`
2	`25`
3	`arg`
4	`1`
5	`con`
6	`1`
7	`<=`
8	`jump_zero`
9	`6`
10	`con`
11	`1`
12	`jump`
13	`12`
14	`arg`
15	`1`
16	`arg`
17	`1`
18	`con`
19	`1`
20	`-`

21	call
22	0
23	*
24	return
25	con
26	10
27	call
28	0

Добра идея е всичките оператори да бъдат съпоставени с уникален номер, за да може целият израз да бъде представен като масив от цели числа.

Забележки: Езикът VILLIE на практика е много малка част от функционалния език SML (Standard Meta Language). Описанието на операциите по-горе е по подобие на езикът L(VM), създаден от проф. д-р Michael Kohlhase.

Автор: Георги Гюрчев