

# Проект по Структури от данни и програмиране

(спец. Компютърни науки, 2017/18 г.)

## Интерпретатор

Разглеждаме учебния език EXPR. Той се състои от два вида оператори: оператори за присвояване и оператори за вход или изход на данни. Програмите на този език са редици от оператори за присвояване или извеждане. На всеки ред на една програма на EXPR може да има точно един от двата типа оператори. Имената на променливите в нашия език са съставени само от малки латински букви. Освен това езикът разполага с цели положителни числа в интервала на допустимите стойности на типа `unsigned long int`. Операторите за присвояване са два вида. Единият вид позволява на дадена променлива да се присвои или число, или стойност на прост аритметичен израз. Другият оператор позволява дефиниране на числова функция на една променлива чрез задаване на прост аритметичен израз в който може да участва променливата на функцията. Операторът за извеждане на резултата позволява извеждането на стойността на израз. Операторът за въвеждане на стойност позволява на дадена променлива да бъде присвоена стойност прочетена от стандартния вход. Всеки ред от нашия език, може да бъде описан със следната контекстно-свободна граматика  $\Gamma = \langle N, T, Line, P \rangle$ , чиито продукционни правила имат вида:

```
Line → Var = Expr | Fun[Var] = Expr | print Expr | read Var |  
Fun → A | B | ... | Z | AFun | BFun | ... | ZFun  
Var → a | b | ... | z | aVar | bVar | ... | zVar  
Num → 0 | ... | 9 | 1Num | ... | 9Num  
Expr → Expr + Term | Expr - Term | Term  
Term → Term * Factor | Term / Factor | Term % Factor | Factor  
Factor → Var | Num | (Expr) | Fun[Expr]
```

Да се напише интерпретатор на езика EXPR. Програмата, написана на EXPR да се прочита от входен текстов файл, след което да се изпълняват последователно операторите от програмата и да се извежда резултата на стандартния изход. Ако интерпретаторът срещне оператор за извеждане на стойност, чиято стойност е недефинирана — например променливата няма присвоена стойност, да се извежда подходящо съобщение. Аналогично, операторът за присвояване да предупреждава при деление на нула. Ако интерпретаторът срещне ред, който не е валиден оператор или израз на езика EXPR, да предупреждава по подходящ начин — например чрез съобщението 'Syntax Error at line #'.

Обърнете внимание, че EXPR работи само с цели положителни числа и затова за операцията деление се подразбира целочислено деление.

## Пример

### Входен файл

```
a = 1  
read b  
c = 3  
D[x] = a + 2*b%x
```

```
print a
print b
print c
print D[c]
```

Изход (допускаме, че от стандартния вход е въведена стойност 2 за променливата b).

```
1
2
3
2
```

## Насоки

Една идея е изразите и операторите да се представят по един и същи начин — чрез дървета. Възлите в тези дървета ще са от няколко типа:

- Цяло число (без наследници);
- Променлива (без наследници);
- Оператор за присвояване (ляв наследник е името на променливата, десен наследник е дърво на аритметичния израз, чиято стойност трябва да се присвои на променливата);
- Оператор за извеждане (един наследник, съдържащ дърво на аритметичния израз, чиято стойност ще се извежда);
- Оператор за въвеждане (един наследник, съдържащ името на променливата, чиято стойност ще се въвежда);
- Двуместна операция (наследници са изразите, описващи аргументите на операцията);
- Оператор за дефиниция на функция (първият наследник е името на функцията, вторият е променливата на функцията, а третият е дърво на аритметичния израз, който представя тялото на функцията);
- Оператор за прилагане на функция (ляв наследник е името на функцията, а десен наследник е дърво на аритметичен израз, който представя аргумента на функцията)

Представянето на аритметични изрази с дървета има предимството, че позволява многократно пресмятане на един и същи израз, който представя тялото на функцията.

### Бонуси:

- да се реализира работа с произволно дълги числа
- да се реализират условни оператори
- да се реализират цикли
- да се реализират рекурсивни функции