Интерпретатор на логически изрази

## Условие

Да се направи програма, позволяваща работа с логически изрази. Програмата трябва да получава параметрите си от командния ред. Тя трябва да позволява изграждане и използване на колекция от логически функции.

Програмата трябва да разполага със следните функционалности:

### 1. Въвеждане на логически функции.

Всяка функция се дефинира като именован логически израз, съставен от базовите логически операции AND, OR и NOT или други въведени от потребителя функции. Въвеждането на логическа функция трябва да проверява за наличие на грешки. Базовите операции трябва да имат предварително определен приоритет като той може да се променя чрез употреба на скоби.

Примери:

DEFINE func1(a, b): "a & b" //дефинира функция 'func1', извършваща логическата операция 'AND' между два операнда 'a' и 'b';

DEFINE func2(a, b, c): "func1(a, b) | c" //дефинира функция 'func2', извършваща логическата операция определена от 'func1' между операндите 'a' и 'b', и логическа операция 'OR' между резултата от 'func1' и операнда 'c'. Функцията 'func1' трябва да е била дефинирана от потребителя преди дефинирането на функция 'func2'.

DEFINE func3(a, b, c, d): "a & (b | c) & !d"

DEFINE func4(a, b, c): "a & b | c | !d" //грешка операторът "d" не е дефиниран

### 2. Запис и четене от файл.

Запис (при дефиниране на нова функция) и четене (при стартиране на програмата) на дефинираните от потребителя функции от файл;

### 3. Решаване на функция за дадени параметри.

Примери:

SOLVE func1(1, 0) -> Result: 0;

SOLVE func2(1, 0, 1) -> Result: 1;

При повторно последователно изпълнение на функция, да се преизчисляват само частите от израза, за които това е необходимо. Например, ако се изчислява резултатът за функцията func2(1, 0, 0) и след това за func2(1, 0, 1), това трябва да доведе до преизчисление само на операцията 'OR', тъй като 'func1' вече е била изчислена за тези аргументи при първото изчисление на func2.

### 4. Изготвяне на таблица на истинност за логическа функция.

Пример:

ALL func1 -> a | b | res

0 | 0 | 0

0 | 1 | 0

1 | 0 | 0

1 | 1 | 1

### 5. Намиране на логическа функция.

Намиране на логическа функция по зададена таблица на истинност (или част от нея), като функция, съставена от базовите логически функции (AND, OR, NOT) и дефинираните от потребителя функции. Може да се използва търсене с пълно изчерпване или друг, предложен от студента подход. Подаването на търсената таблица на истинност може да става през потребителския интерфейс:

Пример:

FIND 0,0,0:0;

0,0,1:0;

0,1,0:0;

0,1,1:0;

1,0,0:0;

1,0,1:0;

1,1,0:0;

1,1,1:1

или чрез фейл, в който последната колона е за резултата, а предходните са за операндите.

Пример:

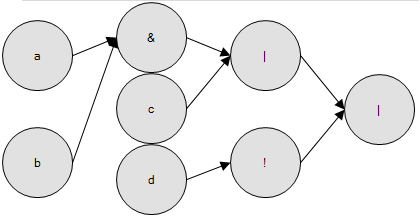
FIND "d:\table.csv"

Result: "a & b & c" или "func1(a, b) & c"

### 6\*. Визуализация на функция.

Визуализация на функция като дървовидна структура чрез графичен интерфейс (GDI), с възможност за показване на изчислената за всеки възел стойност.

Пример:



Фиг. 1. Визуализация на функцията като дървовидна структура.

## Реализация и точки

Всички функции за обработка на текст трябва да се реализират от студента (не е разрешено използването на функциите string.Split, string.IndexOf, Regex, функциите на LINQ и т.н.). Всички помощни структури и типове трябва да се реализират от студента, в това число стекове, свързани списъци, хеш таблици, дървета и т.н.

Студентът трябва да реализира програмата в следната задължителна последователност:

1. Въвеждане на логически функции.   
    Допустимо е изразите да се въвеждат чрез обратен полски запис или както са показани в примерите.

Макс. брой точки за реализация: 20;

1. Запис и четене от файл.

Функциите за запис и четене трябва да записват информацията в текстов (четим) вид.

Макс. брой точки за реализация: 5;

1. Решаване на функция за дадени параметри.

Студентът трябва да реализира разнасяне на изчислените стойности от листата на дървото (операндите на израза), към корена (крайният резултат). Изчислението трябва да бъде рекурсивно.

Макс. брой точки за реализация: 10;

1. Изготвяне на таблица на истинност за логическа функция.

Не е разрешено използването на готови функции за генериране на вариации за операндите.

Макс. брой точки за реализация: 10;

1. Намиране на логическа функция.

Ако търсенето се реализира чрез пълно изчерпване, трябва да се реализират всички възможни вариации на дърво с един възел, с два възела и тн, стига дърветата да са с коректен брой операнди.

Макс. брой точки за реализация: 15;

1. \* - Тази точка е незадължителна. При реализиране на всички точки, вкл. незадължителната, студентът ще бъде освободен от изпит с отлична оценка.