G = ({E, E’, T, T’, F}, {+, \*, (, ), i}, P, E)

S -> E#

E -> TE’

E’ -> +TE’ | e

T -> FT’

T’ -> \*FT’ | e

F -> (E) | i

A nyelvtan segítségével a következő kifejezések validálhatók:

The following grammar is able to decide if the expressions like this: ( 3 \* 3 ) + 2

is correct or not:

( 3 \* 3 ) + 2

A kifejezéseket egyszerűsíteni kell a következő formában:

The terms should be simplified as follows:

pl.: ( 3 \* 4 ) + 2 kifejezésből az ( i + i ) \* i egyszerűsített formulát kell létrehozni a következő reguláris kifejezéssel:

eg. from the expression (3 \* 4) + 2 the simplified formula (i + i) \* i must be created with the following regular expression:

expressionstring = Regex.Replace(expressionstring, ”[0-9]+”, ”i” );

i + ( i \* i ), i + i, i \* i, ...

A szabályrendszer alkalmazásához az alábbi táblázatot kell használni.

A képen asztal látható

Automatikusan generált leírás

A program a következő módon működik:

The program works as follows:

Az algoritmus

The algorithm

1. Szükség van egy input szalagra, ami egy string típusú változó és egy int típusú változóra az input indexeléséhez.   
   You need an input, which is a variable of type string, and a variable of type int to index the input.
2. A táblázatot tárolni kell egy StringGrid, vagy hasonló típusú kontrollban. A tárolás történhet grafikusan, vagy egy n\*m-es string mátrixban.  
   2. The table must be stored in a StringGrid or similar type of control. Storage can be done graphically or in an n \* m string matrix.
3. Szükség van egy Stack típusú változóra. Az elemzés során ebbe a verembe terminális és nemterminális jelek kerülnek bele. A verem kezdetben a szabályrendszer start szimbólumát tartalmazza (E).  
   A variable of type Stack is required. During the analysis, terminal and non-terminal simbols are stored in this stack. The stack initially contains the start symbol of the ruleset (E).
4. Az input változó keztdetben a teljes elemzendő kifejezést tartalmazza. Pl.: String input = ”(i+i)\*i#”. Az input kifejezés végére egy # jelet kell elhelyezni.  
   The input variable contains the complete expression to be analyzed. Eg: String input = ”(i + i) \* i #”. A # sign must be placed at the end of the input expression.
5. A működési ciklus a következő: Be kell olvasni a soron következő elemet (minden lépésben az i index alapján az aktuális karaktert) az input szalagról.  
   The program cycle is as follows: The actual element (the current character based on the i index in each step) must be read from the input string.
6. Ki kell venni a verem legfelső elemét (POP művelet).  
   Must read the top element of the stack (POP operation).
7. Az így kapott adatokat sor és oszlopindexnek kell használni a szabályokat tartalmazó mátrix indexeléséhez.  
   The data thus obtained should be used as a row and column index to index the matrix containing the rules.
8. Az így azonosított cellából vegyük ki az ott található elemet (a mátrixnak az az eleme, amit a két kiolvasott elem alapján azonosítunk).  
   Read the element from the cell thus identified (the element of the matrix that is identified by the two previous components).
9. Pl.: ha az input string aktuális eleme a +, a veremben pedig az E’ nemterminális elem található, akkor az azonosított cella a (+TE’, 2), ahol a vessző bal oldalán egy szabály jobb oldala, a vessző jobb oldalán pedig a szabály sorszáma található.  
   9. For example, if the current element of the input string is + and the stack contains the non-terminal element E ', then the identified cell is (+ TE', 2), where the left side is a rule and the right side containes the serial number of the applied rule.
10. Az azonosított cellában 4 féle elem fordulhat elő.  
    There are 4 types of items that can occur in an identified cell.
    1. Ha a cella üres, az azt jelenti, hogy a kifejezésben hibát találtunk.  
       If the cell is empty, we have found an error in the expression.
    2. Ha a cella az elfogad szót tartalmazza, akkor a végére értünk az elemzésnek, és a kifejezés helyes.  
       If the cell contains the word „accept” (elfogad), we have reached the end of the analysis and the expression is correct.
    3. Ha a pop szó található a cellában, akkor el kell távolítani a verem tetején található elemet (egy karaktert, ami lehet terminális, vagy nemterminális jel), és az indexet léptetni kell, vagyis megnövelni az index változó értékét egyel.  
       If the word pop is in the cell, you must remove the element at the top of the stack (a character, which can be a terminal or non-terminal simbol) and eed to increase the value of the index variable by one.
    4. Ha a cella egy zárójeles szabályt tartalmaz, akkor:  
       If the cell contains a rule in parentheses, then:
       1. el kell távolítani a zárójeleket.  
          the parentheses must be removed.
       2. a vessző bal oldalán található szabályt és a jobb oldalán található sorszámot be kell tenni egy-egy változóba (pl.: a string[] elemek = String.Split(elemek, ”,”) metódussal).  
          the rule to the left of the comma and the number to the right of the comma must be inserted into a variable (for example, with the string [] elements = String.Split (elements, “,”) method).
       3. A vessző bal oldalán található szabályt karakterenként a verembe kell helyezni.  
          The rule to the left of the comma must be placed in the stack character by character.
       4. A szabály sorszámát el kell tárolni egy listába. Ezt az adatot nem használjuk, de a segítségével elő lehetne állítani a program szintaxisfáját.  
          The sequence number of the rule must be stored in a list. We do not use this data, but it could be used to generate the syntax tree of the program.
11. Az 5-10 lépéseket addig kell ismételni, amíg el nem érjük az input végét, vagy hibát nem találunk.  
    Steps 5-10 should be repeated until the end of the input is reached or an error is found.
12. Amennyiben a működési ciklus végén a verem üres, és az input szalag végére értünk, a kifejezés helyes.  
    If the stack is empty at the end of the operating cycle and the end of the input is reached, the expression is correct.
13. Ezt úgy is megállapíthatjuk, ha a két változó által azonosított cella a szabályokat tartalmazó mátrixban az elfogad szót tartalmazza.  
    This fact can also be determined if the cell identified by the two variables contains the word „accept” (elfogad) in the matrix containing the rules.

A programnak folyamatosan mutatnia kell a lépéseket egy rendezett hármasban (tuple, ordered triplet) a következő formában:

The program must continuously show the steps in a ordered triplet (tuple) in the following form:

(input, stack, serial number of the rules)

eg.

( i+i\*i#, E, emptylist ) initially

ezután minden lépésben:

then in each step:

( +i\*i)#, +TE’#, 14862)

where:

* az első elem az aktuális input szalag maradék része,
* a középső elem a verem aktuális tartalma,
* a jobboldali elem pedig az eddig alkalmazott szabályok sorozata.

• the first element is the last part of the current input,

• the middle element is the current content of the stack,

• the right-hand element is a series of rules applied so far.

A programot megvalósíthatjuk grafikus felület használatával, vagy console application formában. A programozási nyelv nincs kikötve de javasolt a C#, vagy a Java nyelvek használata.

The program can be implemented using a graphical interface or in the form of a console application. The programming language is not specified but it is recommended to use C# or Java languages.

Meg kel oldani, hogy a szabályokat tartalmazó táblázat alapértelmezetten tartalmazza a minta táblázatot, de az elemeit a felhasználó tudja szerkeszteni.

It should be resolved that the table containing the rules includes the sample table by default, but its elements can be edited by the user.

Az input kifejezést a felhasználó tudja megadni. Ha a felhasználó a kifejezés végére nem ír # jelet, akkor a program ezt pótolja helyette.

The input expression (what we want to examine) can be specified by the user. If the user does not write a # sign at the end of the expression, it is replaced instead.

Egy példa a program levezetésére:

An example of deriving a program:

A képen asztal látható

Automatikusan generált leírás

A képen asztal látható

Automatikusan generált leírás

( i+i\*i#, E#, e)

( i+i\*i#, TE’#, 1)

( i+i\*i#, FT’E’#, 14)

( i+i\*i#, iT’E’#, 148)

( +i\*i#, T’E’#, 148)

( +i\*i#, E’#, 148)

( +i\*i#, +TE’#, 1482)...

( #, E’#, 1486248586)...

(#, #, ......)

Az input szalag tartalmazza az i+i\*i kifejezést. A veremben az E szimbólum található. Ez a szabályrendszer start szimbóluma.

The input string contains the expression i + i \* i. The E symbol is in the stack. This is the start symbol of the rule system.

Kivesszük a legfelső elemet a veremből. Ez az E nemterminálisjel. Kiolvassuk at index által mutatott karaktert (az első karakter az első lépésben). Ezt a két elemet úgy használjuk a táblázatban, mint sor és oszlopindex. A két index a következő cellát azonosítja: (TE’, 1).

We take the top element out of the stack. This is the nonterminal simbol E. Read the character indicated by index (the first character in the first step). These two elements are used in the table as row and column index. The two indexes identify the following cell: (TE ’, 1).

A képen asztal látható

Automatikusan generált leírás



A zárójeleket eltávolítjuk. A vessző bal oldalán található karaktereket egyesével a verembe helyezzük. Így a rendezett hármas tartalma a következő lesz:

( i+i\*i#, TE’#, 1), mivel az input nem változik, a veremből töröljük a már indexelésre használt legfelső elemet, helyére beírjuk a cellából vett szabály karaktereit, és felírjuk az alkalmazott szabály sorszámát.

The parentheses are removed. The characters to the left of the comma are placed one at a time on the stack. Thus, the contents of the ordered triple will be as follows:

(i + i \* i #, TE ’#, 1), since the input does not change, we delete the topmost element already used for indexing from the stack, replace it with the characters of the rule taken from the cell, and write down the sequence number of the applied rule.

A következő lépésben újra kivesszük a verem legfelső elemét, és az index által mutatott elemet az input string-ből. Ez jelenleg még az első elem, mert az indexet nem léptettük. A verem tetején most a T nemterminális található.

In the next step, we remove the topmost element of the stack and the element shown by the index from the input string again. This is still the first item at this time because we did not increase the index. At the top of the stack is now the non-terminal T.

A két elem által mutatott cella a következő: (FT’, 4). Megismételjük az előző lépéseket.

Ha a cella nem szabályt, hanem a pop kifejezést tartalmazza, akkor növeljük az index értékét, és töröljük a verem legfelső elemét.

The cell indicated by the two elements is: (FT ’, 4). Repeat the previous steps.

* Ha a cella üres, akkor hibát találtunk, megállíthatjuk az elemzést.

If the cell contains not the rule but the expression pop, we increase the value of the index and delete the top element of the stack.

* Ha a cella az elfogad szót tartalmazza, akkor megállhatunk, a kifejezés helyes.  
  If the cell contains the word „accept” elfogad, you can stop, the expression is correct.
* Ha egy szabály az epsylon (e) kifejezést tartalmazza, akkor ezt nem kell a verembe írni.  
  If a given rule contains the expression epsylon (e), it does not need to be written to the stack.

A lépéseket ismételve a fenti levezetést kapjuk. A program megírása előtt papíron, tollal, vagy ceruzával érdemes az elemzést levezetni.

Repeating the steps gives the above derivation. Before writing the program, it is advisable to conduct the analysis on paper, pen or pencil.

You can find more information about this program here:

<https://regi.tankonyvtar.hu/hu/tartalom/tamop425/0046_algorithms_of_informatics_volume1/ch02s03.html>

This chapter should be read before writing the program! Highly recommended!