鬼自摸(おにづも)

目的

C++言語もしくはC言語による正規表現ライブラリの実現

条件

Borland C++でコンパイルすること(ANSI C)
無限後退はしない(再帰的に関数を使わない)
DotNetおよび類似の言語は使わない(言語仕様に正規表現が含まれてしまう)
Yet Another C-CompilerおよびLEXは使わない(これらの使用は無限後退である)

背景

過去にC++言語による実装を試みたが、関数に再帰の限界数(30-40程度)があるため断念
(関数呼び出しのスタックオーバーフローで異常終了した)
鬼車による実装があるが、LEAL(=id.)もしくは再帰的に関数を実行していると思われる

計画

言語の使い方、意味論および内容が、既存のコードとは根本的に違うと思われる。
関数ポインタ(高階関数)は必要だろう。
setjmp, longjmp(浅い方への脱出しかできない) あるいは VM あるいは 中間言語(複雑性が増す?)
等、幅広く検討する。
一致に失敗したときにnグラム(n>0)戻る機能“fall-back”は必須である。多段ループ?難?
多段ループを動的にインライン展開できる?⇒インタプリタなら可?⇒ハツシユをキーに関数呼び出し
省力化のためスタックの自前操作が必要かもしれない⇒inline assembler?(BCCにはない)
普通(今まで通り)に書くと躓く。革新的なアイデアを練ってから
Ajailのように少しづつ、簡単な記号(Alphabets)や簡単な量化(quantify)から始める。

切り株

WindowsのGraphのように小関数を繋げたり、付け替えたりする。
分岐 ([a-z]+|ABc|123) は不可能⇒あきらめる⇒全体として逐次処理のみ可能(1次元)自由度1

pseudo code // quantify 量化1

[a-c1-3]*

```
targetStr='abc123';
int[] lambda(ptr,targetStr){
 yield 0; // * 0-char must match!
 for(int i=0;ptr[i]!=NULL;i++){
  if(ptr[i] in targetStr){
   yield i; // forms rest of array[int]
  }else{
   return -1; // end of array is -1
  }
 }
}
```

pseudo code // quantify 量化2

[C-R]+

```
targetStr='CDEFGHIJKLMNOPQR';
int[] lambda(ptr,targetStr){
 for(int i=0;ptr[i]!=NULL;i++){
  if(ptr[i] in targetStr){
   yield i; // forms array[int]
  }else{
   return -1; // end of array is -1
  }
 }
}
```

pseudo code // sequencial 逐次

AbC

```
targetStr='AbC'
int[] lambda(ptr,targetStr){
 for(int i=0;i<_tcslen(targetStr);i++){
  if(ptr[i]==targetStr[i]){
   // nothing to do
  }else{
   return [-1]; // not match
  }
 }
 return [_tcslen(targetStr),-1]; // match
}
```

このような小関数をたくさん並べて、
投機的実行の分岐予測に選ばせる。オラクル