




水野修(@omzn)

全体の説明







言語処理プログラミング

-  C言語を用いて、比較的簡単なプログラミング言語のコンパイラを作成する
-  コンパイラの基本的な構造とテキスト処理の手法を理解する
-  比較的大きなプログラムを作成する経験を得る





構成とスケジュール


 課題 1: 字句の出現頻度表の作成

 期間 2024-09-30(月) ~ 2024-10-28(月)

 課題 2: プリティプリンタの作成

 期間 2024-10-21(月) ~ 2024-11-25(月)

 課題 3: クロスリファレンサの作成

 期間 2024-11-19(月) ~ 2024-12-16(月)






 課題 4: コンパイラの作成

 期間 2024-12-09(月) ~ 2025-01-27(月)






レポート

1. 作成したプログラムの設計情報

-  (a)全体構成:どのようなモジュールがあるか, それらの依存関係 (呼び出し関係 やデータ参照関係)
 -  (b)各モジュールごとの構成: 使用されているデータ構造の説明と各変数の意味
 -  (c)各関数の外部 (入出力) 仕様: その関数の機能.引数や戻り値の意味と参照する大域変数, 変更する大域変数などを含む
-  (注) ただし, それ以前のレポートに記載した内容と同じである場合には, その旨のみを記述するだけでよい. たとえば, 関数〇〇が課題 1で説明済みであり何も変更されていなければ, 関数〇〇:課題 1 レポートで記載済みと書くだけでよい.
-  なお, モジュールとは, 意味的にまとまったプログラムの部分である. 小さなプログラムの場合は関数一つ一つをモジュールと考える場合もあるし, いくつかの大域変数を共有して使う関数群を (その大域変数も含めて) モジュールとする場合もある. たとえば, 課題1で作成する字句解析系はモジュールである.

2. テスト情報

-  (a)テストデータ (既に用意されているテストデータについてはファイル名のみでよい)
-  (b)テスト結果 (テストしたすべてのテストデータについて)
-  (c)テストデータの十分性 (それだけのテストでどの程度バグがないことが保証できるか) の説明




レポート

2. テスト情報

(a) テストデータ

 既に用意されているテストデータについてはファイル名のみでよい

(b) テスト結果

 テストしたすべてのテストデータについて

(c) テストデータの十分性

 それだけのテストでどの程度バグがないことが保証できるか

 こちらが提供しているテストプログラムで十分である保証は無い




レポート(続き)

3. この課題を行うための事前計画 (スケジュール) と実際の進捗状況

 Redmine を利用する場合は、この章は省略できる。


(a)事前計画 (スケジュール)

 当初の計画と、演習中に計画を大きく修正した場合には修正後の計画 (最終分だけでよい) を記載すること。計画を立て、×切までに完成したプログラムとレポートを提出するまでが演習である。

(b)事前計画の立て方についての前課題からの改善点 (課題1を除く)

(c)実際の進捗状況







(d)当初の事前計画と実際の進捗との差の原因

 事前計画と進捗状況は、開始 (予定) 日、終了 (予定) 日、使用 (見積もり) 時間、作業 (予定) 内容の 4 項目をカラムとし行は日付順とする表形式で記述すること (実務では線表で書かれる)。

 作業 (予定) 内容は 1 行程度でよい。詳細な説明は課題1の付録参照。










レポートの提出

-  PDF もしくは、Word の形式のファイル.
 -  A4の用紙サイズで、表紙 (1 ページ目) に課題名 (「言語処理プログラミング 課題 1」 など)、提出日の日付、学籍番号、氏名のみを記載
-  提出先
 -  Redmine に作成してある個人用のプロジェクトの「レポート」チケットへ添付
-  提出期間
 -  各課題提出期間




プログラムの提出

-  プログラムもRedmine上の個人別プロジェクト内にある該当する「レポート」チケットに提出する.
-  プログラムを提出する旨の指示が書いてあります.
-  提出物
 -  基本的には以下:
 -  ソースプログラムファイルのみ (*.c, *.h のファイル)
 -  それらを同一のフォルダ (ディレクトリ) に置いて, まとめてコンパイルすれば, 実行形式ファイルが生成されるものを提出
 -  makeを用いることができるようにMakefileなどを記述しても良い



プログラムの提出


 実行ファイル名:
 課題1 は「tc」, 課題2 は「pp」, 課題3 は「cr」, 課題4 は「mpplc」


 以下の方法のいずれかで, 上に指定する実行ファイルが生成できるようにしてください.

 `gcc -o 実行ファイル名 *.c`

 または

 `make`


 `make` を使う場合は, Makefile も必ず提出ファイルに含めてください.

 ファイルは一식을zipで固めて提出してください.





Docker

 演習室環境と同じ開発環境を再現するDockerイメージを配布


 無保証

 自動テストプログラムも格納している

 インストール: Moodleを参照







 ここの手順だけではうまくいかないこともある

 dockerの使い方を別途習得してから試そう

 Dockerを使わなくても演習は実施可能なので、あまりここで詰まらな
いで欲しい



テスト環境 lpp_test

-  Moodleを参照
-  lpp_testは本課題専用のテスト環境
 -  ブラックボックステスト
-  課題1～4について，提供しているサンプルを用いてテストを実施
 -  Dockerイメージにも同梱している
 -  Pythonの環境を構築すれば，自力でも実行できるはず



ホワイトボックステスト gcov



テキスト1.8.4節を参照




gcovを活用することで、実行していない行・分岐がどれぐらいあるかを確認できる。





書きっぱなし，を防止できる。


スケジューリング

Redmineの利用を推奨

 最初に**題名**で**昇順**に**ソート**してください。（「**題名**」を何回か押す）


 Redmineに開発の記録を残しておけば，レポートへの記載は不要とする


 実態ベースで記録すること


 後から「**終了**」だけ押して終わらせるのはあまり評価しない


<input type="checkbox"/>	#	トラッカー	ステータス	題名 ^	開始日	期日
<input type="checkbox"/>	18350	言プロ課題	新規	01 課題1: 字句の出現頻度表の作成	2024/09/30	2024/10/30
<input type="checkbox"/>	18351	設計	新規	> 01.01 スケジュールを立てる	2024/09/30	
<input type="checkbox"/>	18352	設計	新規	> 01.02 資料を読む		
<input type="checkbox"/>	26810	設計	新規	> 01.02.01 配布された資料を読み直す		
<input type="checkbox"/>	26812	設計	新規	> 01.02.02 コンパイラのテキスト (プログラム) を読む		
<input type="checkbox"/>	26811	設計	新規	> 01.02.02 配布されたプログラムを読み直す		
<input type="checkbox"/>	18353	設計	新規	> 01.03 字句解析系 (スキャナ) の概略設計		
<input type="checkbox"/>	18354	コーディング	新規	> 01.04 プログラム作成 (コーディング)		
<input type="checkbox"/>	26813	コーディング	新規	> 01.04.01 トークンカウント用の配列初期化の作成		
<input type="checkbox"/>	26814	コーディング	新規	> 01.04.02 トークンをカウント部分の作成 (スキャナを利用)		
<input type="checkbox"/>	26815	コーディング	新規	> 01.04.03 カウントした結果の出力部分の作成		
<input type="checkbox"/>	26816	コーディング	新規	> 01.04.04 スキャナの作成		
<input type="checkbox"/>	18359	テスト	新規	> 01.05 動作テストの実施		
<input type="checkbox"/>	26817	テスト	新規	> 01.05.01 ブラックボックステスト用プログラムの作成		
<input type="checkbox"/>	26818	テスト	新規	> 01.05.02 ホワイトボックステスト用プログラムの作成		
<input type="checkbox"/>	26819	テスト	新規	> 01.05.03 テストとデバッグを行う		
<input type="checkbox"/>	21190	提出	新規	> 01.06 プログラムの提出	2024/10/21	2024/10/30
<input type="checkbox"/>	24054	提出	新規	> 01.07 レポートの提出	2024/10/21	2024/10/30
<input type="checkbox"/>	21177	言プロ課題	新規	02 課題2: プリティプリンタの作成	2024/10/21	2024/11/25
<input type="checkbox"/>	21178	設計	新規	> 02.01 スケジュールを立てる	2024/10/21	
<input type="checkbox"/>	21179	設計	新規	> 02.02 資料を読む		
<input type="checkbox"/>	21180	設計	新規	> 02.03 構文解析系 (パーサー) の概略設計		
<input type="checkbox"/>	21181	コーディング	新規	> 02.04 プログラム作成 (コーディング)		
<input type="checkbox"/>	21186	テスト	新規	> 02.05 テストの実施		
<input type="checkbox"/>	21415	提出	新規	> 02.06 プログラムの提出	2024/11/19	2024/11/25
<input type="checkbox"/>	24055	提出	新規	> 02.07 レポートの提出	2024/11/19	2024/11/25
<input type="checkbox"/>	21416	言プロ課題	新規	03 課題3: クロスリファレンサの作成	2024/11/19	2024/12/16
<input type="checkbox"/>	21417	設計	新規	> 03.01 スケジュールを立てる	2024/11/19	

スケジュールリング

- 

大きく4つの「言プロ課題」がある.
- 

その中に「設計」「コーディング」「テスト」「提出」の子チケットがある.
- 

子チケットが全て終了すると言プロ課題も終了できる. (手動で終了させる)
- 

「提出」チケットは審査待ちになったものを受理できたら教員が終了させる.



01 課題1: 字句の出現頻度表の作成

水野 修 が1年以上前に追加. 9分前に更新.

ステータス:	新規	開始日:	2024/09/30
優先度:	通常	期日:	2024/10/30
担当者:	-	進捗率:	<div></div> 0%
		予定工数:	(合計: 0:00時間)

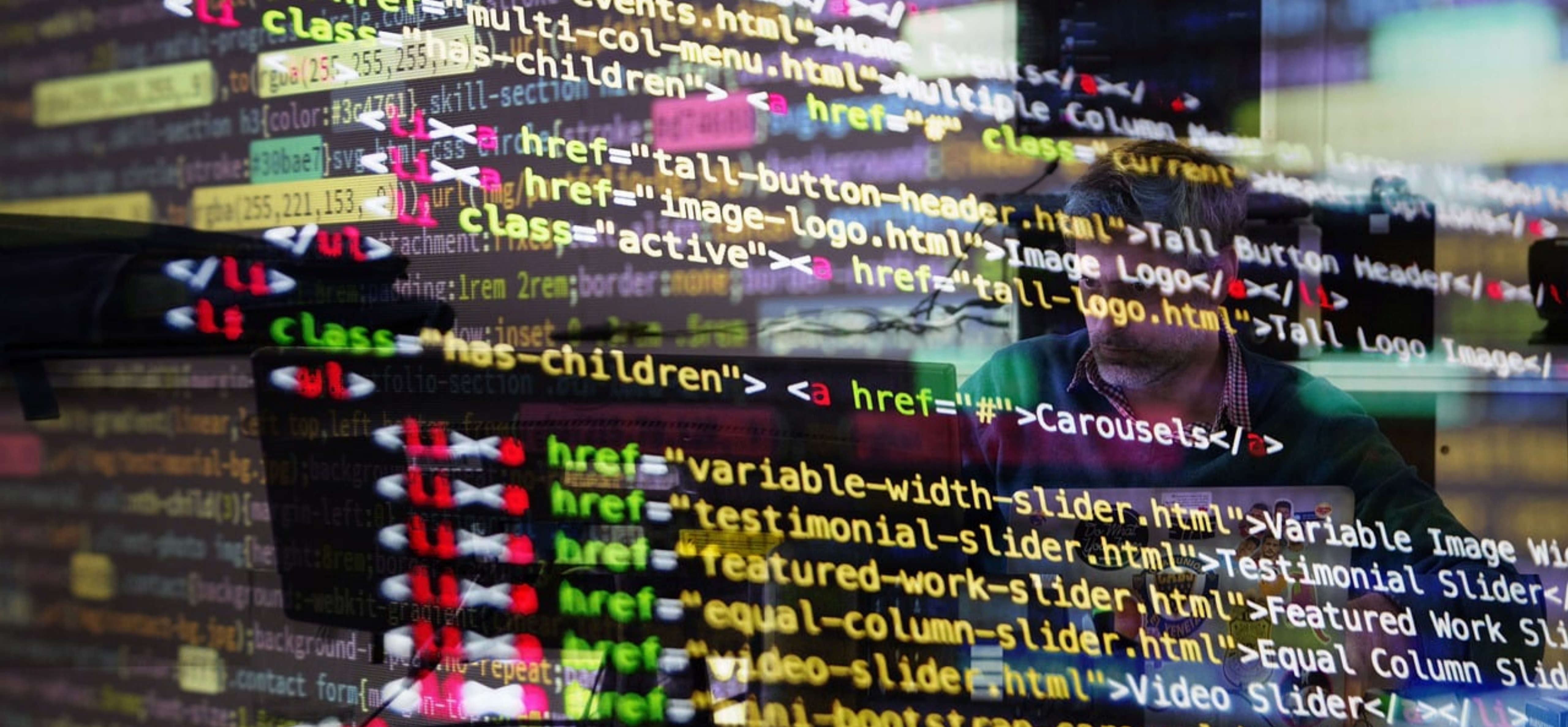
ステータス--> [新規] [進行中]

説明

言語処理プログラミング課題1

子チケット

設計 #18351: 01.01 スケジュールを立てる	新規	<div></div>
設計 #18352: 01.02 資料を読む	新規	<div></div>
> 設計 #26810: 01.02.01 配布された資料を読み直す	新規	<div></div>
> 設計 #26811: 01.02.02 配布されたプログラムを読み直す	新規	<div></div>
> 設計 #26812: 01.02.02 コンパイラのテキスト (プログラム) を読む	新規	<div></div>
設計 #18353: 01.03 字句解析系 (スキャナ) の概略設計	新規	<div></div>
コーディング #18354: 01.04 プログラム作成 (コーディング)	新規	<div></div>
> コーディング #26813: 01.04.01 トークンカウント用の配列初期化の作成	新規	<div></div>
> コーディング #26814: 01.04.02 トークンをカウント部分の作成 (スキャナを利用)	新規	<div></div>
> コーディング #26815: 01.04.03 カウントした結果の出力部分の作成	新規	<div></div>
> コーディング #26816: 01.04.04 スキャナの作成	新規	<div></div>
テスト #18359: 01.05 動作テストの実施	新規	<div></div>
> テスト #26817: 01.05.01 ブラックボックステスト用プログラムの作成	新規	<div></div>
> テスト #26818: 01.05.02 ホワイトボックステスト用プログラムの作成	新規	<div></div>
> テスト #26819: 01.05.03 テストとデバッグを行う	新規	<div></div>
提出 #21190: 01.06 プログラムの提出	新規	<div></div>
提出 #24054: 01.07 レポートの提出	新規	<div></div>



水野修(@omzn)


課題1の説明




課題1 - 字句解析

 課題名

 字句の出現頻度表の作成

 演習期間






 2024-09-30 ~ 2024-10-20

 レポート・プログラム提出期間

 2024-10-21 ~ 2024-10-28




課題内容


-  プログラミング言語 MPPL で書かれたプログラムらしきものを読み込む.
-  字句 (トークン) がそれぞれ何個出現したかを数え, 出力するCプログラムを作成する.
-  例えば, 作成するプログラム名を tc, MPPL で書かれたプログラムらしきもののファイル名を foo.mpl とするとき, コマンドラインからのコマンドを
-  \$./tc foo.mpl
-  とすれば (foo.mpl のみが引数), foo.mpl 内の字句の出現個数を出力する





入力

MPPL で書かれたプログラムらしきものの ファイル名

 字句 (トークン) は, 名前, キーワード, 符号なし
整数, 文字列, 記号のいずれ

 ただし, キーワードと記号については, それぞれ
の記号列が別々の字句であるが, 名前, 符号なし
整数, 文字列については, その実体が異なってい
ても同じ「名前」, 「符号なし整数」, 「文字列」
という字句であるとする

 字句の定義として, MPPL のプログラムのマイク
ロ構文を右図のように与える (左辺の括弧の中は
非終端記号の英語表記であり参考のために付加し
てある)

 なお, 終端記号はASCII文字である.

プログラム (**program**) ::= { 字句 | 分離子 }

字句 (**token**) ::= 名前 | キーワード | 符号なし整数 | 文字列 | 記号

名前 (**name**) ::= 英字 { 英字 | 数字 }

キーワード (**keyword**) ::= "p" "r" "o" "g" "r" "a" "m" | "v" "a" "r" |
"a" "r" "r" "a" "y" | "o" "f" | "b" "e" "g" "i" "n" | "e" "n" "d" |
"i" "f" | "t" "h" "e" "n" | "e" "l" "s" "e" |
"p" "r" "o" "c" "e" "d" "u" "r" "e" | "r" "e" "t" "u" "r" "n" |
"c" "a" "l" "l" | "w" "h" "i" "l" "e" | "d" "o" | "n" "o" "t" |
"o" "r" | "d" "i" "v" | "a" "n" "d" | "c" "h" "a" "r" |
"i" "n" "t" "e" "g" "e" "r" | "b" "o" "o" "l" "e" "a" "n" |
"r" "e" "a" "d" | "w" "r" "i" "t" "e" | "r" "e" "a" "d" "l" "n" |
"w" "r" "i" "t" "e" "l" "n" | "t" "r" "u" "e" |
"f" "a" "l" "s" "e" | "b" "r" "e" "a" "k"

符号なし整数 (**unsigned integer**) ::= 数字 { 数字 }

文字列 (**string**) ::= "'" { 文字列要素 | "'" "" } "'"

"'"はアポストロフィ (シングルクォート) である

記号 (**symbol**) ::= "+" | "-" | "*" | "=" | "<" ">" | "<" | "<" "=" |
">" | ">" "=" | "(" | ")" | "[" | "]" | ":" "=" | "." | "," |
":" | ";"

英字 (**alphabet**) ::= "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" |
"i" | "j" | "k" | "l" | "m" | "n" | "o" | "p" | "q" | "r" | "s" |
"t" | "u" | "v" | "w" | "x" | "y" | "z" |
"A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" | "I" | "J" | "K" |
"L" | "M" | "N" | "O" | "P" | "Q" | "R" | "S" | "T" | "U" | "V" |
"W" | "X" | "Y" | "Z"

数字 (**digit**) ::= "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" |
"8" | "9"

分離子 (**separator**) ::= 空白 | タブ | 改行 | 注釈

注釈 (**comment**) ::= "{" { 注釈要素 } "}" | "/" "*" { 注釈要素 } "*" "/"



マイクロ構文



構文において、用いられている記号の意味は次の通りである.



" " 終端記号であることを示す



::= 左辺の非終端記号が右辺で定義されることを示す



| 左側と右側のどちらかであることを示す



{ } 内部を 0 回以上繰り返すことを表す



[] 内部を省略してもよい (0 回か 1 回) ことを示す

プログラム (program) ::= { 字句 | 分離子 }

字句 (token) ::= 名前 | キーワード | 符号なし整数 | 文字列 | 記号

名前 (name) ::= 英字 { 英字 | 数字 }

キーワード (keyword) ::= "p" "r" "o" "g" "r" "a" "m" | "v" "a" "r" |
"a" "r" "r" "a" "y" | "o" "f" | "b" "e" "g" "i" "n" | "e" "n" "d" |
"i" "f" | "t" "h" "e" "n" | "e" "l" "s" "e" |
"p" "r" "o" "c" "e" "d" "u" "r" "e" | "r" "e" "t" "u" "r" "n" |
"c" "a" "l" "l" | "w" "h" "i" "l" "e" | "d" "o" | "n" "o" "t" |
"o" "r" | "d" "i" "v" | "a" "n" "d" | "c" "h" "a" "r" |
"i" "n" "t" "e" "g" "e" "r" | "b" "o" "o" "l" "e" "a" "n" |
"r" "e" "a" "d" | "w" "r" "i" "t" "e" | "r" "e" "a" "d" "l" "n" |
"w" "r" "i" "t" "e" "l" "n" | "t" "r" "u" "e" |
"f" "a" "l" "s" "e" | "b" "r" "e" "a" "k"

符号なし整数 (unsigned integer) ::= 数字 { 数字 }

文字列 (string) ::= "" { 文字列要素 | "'" "" } ""

"" はアポストロフィ (シングルクォート) である

記号 (symbol) ::= "+" | "-" | "*" | "=" | "<" ">" | "<" | "<" "=" |
">" | ">" "=" | "(" ")" | "[" "]" | ":" "=" | "." | "," |
":" | ";"

英字 (alphabet) ::= "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" |
"i" | "j" | "k" | "l" | "m" | "n" | "o" | "p" | "q" | "r" | "s" |
"t" | "u" | "v" | "w" | "x" | "y" | "z" |
"A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" | "I" | "J" | "K" |
"L" | "M" | "N" | "O" | "P" | "Q" | "R" | "S" | "T" | "U" | "V" |
"W" | "X" | "Y" | "Z"

数字 (digit) ::= "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" |
"8" | "9"

分離子 (separator) ::= 空白 | タブ | 改行 | 注釈

注釈 (comment) ::= "{" { 注釈要素 } "}" | "/" "*" { 注釈要素 } "*" "/"



文字列要素 (string element)

- 👤 アポストロフィ`"`，改行以外の任意の表示文字を表す．
- 👤 文字列には長さの概念があり，その文字列に含まれるアポストロフィ`"`，改行以外の任意の表示文字の数と連続する`"` `"`の組の数の和である．即ち`"`については連続する2つを1と数える（同様に連続する4つを2と数える．以下同じ）．言い換えると，文字列のマイクロ構文での`{ }`の繰り返し回数が文字列の長さである．この文字列の長さの概念は，課題3の制約規則において利用する．

```
プログラム (program) ::= { 字句 | 分離子 }
字句 (token) ::= 名前 | キーワード | 符号なし整数 | 文字列 | 記号
名前 (name) ::= 英字 { 英字 | 数字 }
キーワード (keyword) ::= "p" "r" "o" "g" "r" "a" "m" | "v" "a" "r" |
    "a" "r" "r" "a" "y" | "o" "f" | "b" "e" "g" "i" "n" | "e" "n" "d" |
    "i" "f" | "t" "h" "e" "n" | "e" "l" "s" "e" |
    "p" "r" "o" "c" "e" "d" "u" "r" "e" | "r" "e" "t" "u" "r" "n" |
    "c" "a" "l" "l" | "w" "h" "i" "l" "e" | "d" "o" | "n" "o" "t" |
    "o" "r" | "d" "i" "v" | "a" "n" "d" | "c" "h" "a" "r" |
    "i" "n" "t" "e" "g" "e" "r" | "b" "o" "o" "l" "e" "a" "n" |
    "r" "e" "a" "d" | "w" "r" "i" "t" "e" | "r" "e" "a" "d" "l" "n" |
    "w" "r" "i" "t" "e" "l" "n" | "t" "r" "u" "e" |
    "f" "a" "l" "s" "e" | "b" "r" "e" "a" "k"
符号なし整数 (unsigned integer) ::= 数字 { 数字 }
文字列 (string) ::= " { 文字列要素 | " " } "
    # " "はアポストロフィ (シングルクォート)である
記号 (symbol) ::= "+" | "-" | "*" | "=" | "<" ">" | "<" | "<" "=" |
    ">" | ">" "=" | "(" | ")" | "[" | "]" | ":" "=" | "." | "," |
    ":" | ";"
英字 (alphabet) ::= "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" |
    "i" | "j" | "k" | "l" | "m" | "n" | "o" | "p" | "q" | "r" | "s" |
    "t" | "u" | "v" | "w" | "x" | "y" | "z" |
    "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" | "I" | "J" | "K" |
    "L" | "M" | "N" | "O" | "P" | "Q" | "R" | "S" | "T" | "U" | "V" |
    "W" | "X" | "Y" | "Z"
数字 (digit) ::= "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" |
    "8" | "9"
分離子 (separator) ::= 空白 | タブ | 改行 | 注釈
注釈 (comment) ::= "{" { 注釈要素 } "}" | "/" "*" { 注釈要素 } "*" "/"
```



文字列

 'abcde'

 「abcde」という文字列


 長さは5

 'abc"de'

 「abc"de」という文字列

 長さは6

 'abc'de'

 「abc」という文字列と、deという名前と文字列の始まりの'。(その後どうなるかは不明)

 'abc

de'

 文字列の途中で改行はダメ

プログラム (program) ::= { 字句 | 分離子 }

字句 (token) ::= 名前 | キーワード | 符号なし整数 | 文字列 | 記号

名前 (name) ::= 英字 { 英字 | 数字 }

キーワード (keyword) ::= "p" "r" "o" "g" "r" "a" "m" | "v" "a" "r" |
"a" "r" "r" "a" "y" | "o" "f" | "b" "e" "g" "i" "n" | "e" "n" "d" |
"i" "f" | "t" "h" "e" "n" | "e" "l" "s" "e" |
"p" "r" "o" "c" "e" "d" "u" "r" "e" | "r" "e" "t" "u" "r" "n" |
"c" "a" "l" "l" | "w" "h" "i" "l" "e" | "d" "o" | "n" "o" "t" |
"o" "r" | "d" "i" "v" | "a" "n" "d" | "c" "h" "a" "r" |
"i" "n" "t" "e" "g" "e" "r" | "b" "o" "o" "l" "e" "a" "n" |
"r" "e" "a" "d" | "w" "r" "i" "t" "e" | "r" "e" "a" "d" "l" "n" |
"w" "r" "i" "t" "e" "l" "n" | "t" "r" "u" "e" |
"f" "a" "l" "s" "e" | "b" "r" "e" "a" "k"

符号なし整数 (unsigned integer) ::= 数字 { 数字 }

文字列 (string) ::= "" { 文字列要素 | "'" "" } ""

""はアポストロフィ (シングルクォート)である

記号 (symbol) ::= "+" | "-" | "*" | "=" | "<" ">" | "<" | "<" "=" |
">" | ">" "=" | "(" | ")" | "[" | "]" | ":" "=" | "." | "," |
":" | ";"

英字 (alphabet) ::= "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" |
"i" | "j" | "k" | "l" | "m" | "n" | "o" | "p" | "q" | "r" | "s" |
"t" | "u" | "v" | "w" | "x" | "y" | "z" |
"A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" | "I" | "J" | "K" |
"L" | "M" | "N" | "O" | "P" | "Q" | "R" | "S" | "T" | "U" | "V" |
"W" | "X" | "Y" | "Z"

数字 (digit) ::= "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" |
"8" | "9"

分離子 (separator) ::= 空白 | タブ | 改行 | 注釈

注釈 (comment) ::= "{" { 注釈要素 } "}" | "/" "*" { 注釈要素 } "*" "/"



注釈要素 (comment element)



注釈が"{", "}"で囲まれているときは、注釈要素は閉じ中括弧"}"以外の任意の表示文字を表す。



"/" "/*", "/*" "/"で囲まれているときは、連続する注釈要素として"/*" "/"が現れないことを除いて任意の表示文字を表す。



注釈が開始されたままEOFを迎えた場合には、EOFまでを注釈として処理する。

```
プログラム (program) ::= { 字句 | 分離子 }
字句 (token) ::= 名前 | キーワード | 符号なし整数 | 文字列 | 記号
名前 (name) ::= 英字 { 英字 | 数字 }
キーワード (keyword) ::= "p" "r" "o" "g" "r" "a" "m" | "v" "a" "r" |
    "a" "r" "r" "a" "y" | "o" "f" | "b" "e" "g" "i" "n" | "e" "n" "d" |
    "i" "f" | "t" "h" "e" "n" | "e" "l" "s" "e" |
    "p" "r" "o" "c" "e" "d" "u" "r" "e" | "r" "e" "t" "u" "r" "n" |
    "c" "a" "l" "l" | "w" "h" "i" "l" "e" | "d" "o" | "n" "o" "t" |
    "o" "r" | "d" "i" "v" | "a" "n" "d" | "c" "h" "a" "r" |
    "i" "n" "t" "e" "g" "e" "r" | "b" "o" "o" "l" "e" "a" "n" |
    "r" "e" "a" "d" | "w" "r" "i" "t" "e" | "r" "e" "a" "d" "l" "n" |
    "w" "r" "i" "t" "e" "l" "n" | "t" "r" "u" "e" |
    "f" "a" "l" "s" "e" | "b" "r" "e" "a" "k"
符号なし整数 (unsigned integer) ::= 数字 { 数字 }
文字列 (string) ::= "'" { 文字列要素 | "'" "''" } "'"
    # "'"はアポストロフィ (シングルクォート) である
記号 (symbol) ::= "+" | "-" | "*" | "=" | "<" ">" | "<" | "<" "=" |
    ">" | ">" "=" | "(" | ")" | "[" | "]" | ":" "=" | "." | "," |
    ":" | ";"
英字 (alphabet) ::= "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" |
    "i" | "j" | "k" | "l" | "m" | "n" | "o" | "p" | "q" | "r" | "s" |
    "t" | "u" | "v" | "w" | "x" | "y" | "z" |
    "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" | "I" | "J" | "K" |
    "L" | "M" | "N" | "O" | "P" | "Q" | "R" | "S" | "T" | "U" | "V" |
    "W" | "X" | "Y" | "Z"
数字 (digit) ::= "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" |
    "8" | "9"
分離子 (separator) ::= 空白 | タブ | 改行 | 注釈
注釈 (comment) ::= "{" { 注釈要素 } "}" | "/" "/*" { 注釈要素 } "/*" "/"
```




注釈要素



```
{comment} /* comment */
```



正しい注釈2つ



```
{comment}}
```



謎の } が最後に見つかったことになる
(エラー)



```
{/*}/*}*/
```



2つの注釈として正しく処理される



```
program aaa;  
{
```



EOFまでコメントとして読み飛ばし
(エラーでは無い)

プログラム (program) ::= { 字句 | 分離子 }

字句 (token) ::= 名前 | キーワード | 符号なし整数 | 文字列 | 記号

名前 (name) ::= 英字 { 英字 | 数字 }

キーワード (keyword) ::= "p" "r" "o" "g" "r" "a" "m" | "v" "a" "r" |
"a" "r" "r" "a" "y" | "o" "f" | "b" "e" "g" "i" "n" | "e" "n" "d" |
"i" "f" | "t" "h" "e" "n" | "e" "l" "s" "e" |
"p" "r" "o" "c" "e" "d" "u" "r" "e" | "r" "e" "t" "u" "r" "n" |
"c" "a" "l" "l" | "w" "h" "i" "l" "e" | "d" "o" | "n" "o" "t" |
"o" "r" | "d" "i" "v" | "a" "n" "d" | "c" "h" "a" "r" |
"i" "n" "t" "e" "g" "e" "r" | "b" "o" "o" "l" "e" "a" "n" |
"r" "e" "a" "d" | "w" "r" "i" "t" "e" | "r" "e" "a" "d" "l" "n" |
"w" "r" "i" "t" "e" "l" "n" | "t" "r" "u" "e" |
"f" "a" "l" "s" "e" | "b" "r" "e" "a" "k"

符号なし整数 (unsigned integer) ::= 数字 { 数字 }

文字列 (string) ::= "'" { 文字列要素 | "'" "''" } "'"

"'"はアポストロフィ (シングルクォート)である

記号 (symbol) ::= "+" | "-" | "*" | "=" | "<" ">" | "<" | "<" "=" |
">" | ">" "=" | "(" ")" | "[" "]" | ":" "=" | "." | "," |
":" | ";"

英字 (alphabet) ::= "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" |
"i" | "j" | "k" | "l" | "m" | "n" | "o" | "p" | "q" | "r" | "s" |
"t" | "u" | "v" | "w" | "x" | "y" | "z" |
"A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" | "I" | "J" | "K" |
"L" | "M" | "N" | "O" | "P" | "Q" | "R" | "S" | "T" | "U" | "V" |
"W" | "X" | "Y" | "Z"

数字 (digit) ::= "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" |
"8" | "9"

分離子 (separator) ::= 空白 | タブ | 改行 | 注釈

注釈 (comment) ::= "{" { 注釈要素 } "}" | "/" "*" { 注釈要素 } "*" "/"



文字等

- 空白 (space), タブ (tab) は, ASCII コードではそれぞれ 0x20, 0x09(16進数表示) であり, C言語上では, ' ', '\t' で表現される.
- 改行 (end of line) は OS によって異なるので, (1)'\r', (2)'\n', (3)'\r' '\n', (4)'\n' '\r' の 4 通りの ASCII コード (列) を一つの改行として扱うこと.ただし, '\r', '\n' は ASCII コードではそれぞれ 0x0D, 0x0A(16進数表示) である.
- 表示文字 (graphic character) とは, タブ, 改行と通常画面に表示可能な文字 (ASCII では 0x20 から 0x7E(16進数表示) までの文字) を意味する.
- 表示文字ではない文字コード (タブ, 改行以外の制御コード) が現れた場合は, 存在しないものとして無視してよい. もちろん, エラーとしてもよい.

```
プログラム (program) ::= { 字句 | 分離子 }
字句 (token) ::= 名前 | キーワード | 符号なし整数 | 文字列 | 記号
名前 (name) ::= 英字 { 英字 | 数字 }
キーワード (keyword) ::= "p" "r" "o" "g" "r" "a" "m" | "v" "a" "r" |
    "a" "r" "r" "a" "y" | "o" "f" | "b" "e" "g" "i" "n" | "e" "n" "d" |
    "i" "f" | "t" "h" "e" "n" | "e" "l" "s" "e" |
    "p" "r" "o" "c" "e" "d" "u" "r" "e" | "r" "e" "t" "u" "r" "n" |
    "c" "a" "l" "l" | "w" "h" "i" "l" "e" | "d" "o" | "n" "o" "t" |
    "o" "r" | "d" "i" "v" | "a" "n" "d" | "c" "h" "a" "r" |
    "i" "n" "t" "e" "g" "e" "r" | "b" "o" "o" "l" "e" "a" "n" |
    "r" "e" "a" "d" | "w" "r" "i" "t" "e" | "r" "e" "a" "d" "l" "n" |
    "w" "r" "i" "t" "e" "l" "n" | "t" "r" "u" "e" |
    "f" "a" "l" "s" "e" | "b" "r" "e" "a" "k"
符号なし整数 (unsigned integer) ::= 数字 { 数字 }
文字列 (string) ::= "'" { 文字列要素 | "'" "" } "'"
    # "" はアポストロフィ (シングルクォート) である
記号 (symbol) ::= "+" | "-" | "*" | "=" | "<" ">" | "<" | "<" "=" |
    ">" | ">" "=" | "(" | ")" | "[" | "]" | ":" "=" | "." | "," |
    ":" | ";"
英字 (alphabet) ::= "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" |
    "i" | "j" | "k" | "l" | "m" | "n" | "o" | "p" | "q" | "r" | "s" |
    "t" | "u" | "v" | "w" | "x" | "y" | "z" |
    "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" | "I" | "J" | "K" |
    "L" | "M" | "N" | "O" | "P" | "Q" | "R" | "S" | "T" | "U" | "V" |
    "W" | "X" | "Y" | "Z"
数字 (digit) ::= "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" |
    "8" | "9"
分離子 (separator) ::= 空白 | タブ | 改行 | 注釈
注釈 (comment) ::= "{" { 注釈要素 } "}" | "/" "*" { 注釈要素 } "*" "/"
```




制約規則



制約規則としては以下がある.



最長一致規則: 字句として, 二つ以上の可能性があるときは最も長い文字の列を字句とする



abc100 という字句は, 「abc100」という名前である. 「abc」という名前と「100」という数字の連続ではない



英大文字と小文字は区別する



キーワードは予約されている: キーワードは名前ではない

```
プログラム (program) ::= { 字句 | 分離子 }
字句 (token) ::= 名前 | キーワード | 符号なし整数 | 文字列 | 記号
名前 (name) ::= 英字 { 英字 | 数字 }
キーワード (keyword) ::= "p" "r" "o" "g" "r" "a" "m" | "v" "a" "r" |
    "a" "r" "r" "a" "y" | "o" "f" | "b" "e" "g" "i" "n" | "e" "n" "d" |
    "i" "f" | "t" "h" "e" "n" | "e" "l" "s" "e" |
    "p" "r" "o" "c" "e" "d" "u" "r" "e" | "r" "e" "t" "u" "r" "n" |
    "c" "a" "l" "l" | "w" "h" "i" "l" "e" | "d" "o" | "n" "o" "t" |
    "o" "r" | "d" "i" "v" | "a" "n" "d" | "c" "h" "a" "r" |
    "i" "n" "t" "e" "g" "e" "r" | "b" "o" "o" "l" "e" "a" "n" |
    "r" "e" "a" "d" | "w" "r" "i" "t" "e" | "r" "e" "a" "d" "l" "n" |
    "w" "r" "i" "t" "e" "l" "n" | "t" "r" "u" "e" |
    "f" "a" "l" "s" "e" | "b" "r" "e" "a" "k"
符号なし整数 (unsigned integer) ::= 数字 { 数字 }
文字列 (string) ::= "'" { 文字列要素 | "'" "''" } "'"
    # "'" はアポストロフィ (シングルクォート) である
記号 (symbol) ::= "+" | "-" | "*" | "=" | "<" ">" | "<" | "<" "=" |
    ">" | ">" "=" | "(" | ")" | "[" | "]" | ":" "=" | "." | "," |
    ":" | ";"
英字 (alphabet) ::= "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" |
    "i" | "j" | "k" | "l" | "m" | "n" | "o" | "p" | "q" | "r" | "s" |
    "t" | "u" | "v" | "w" | "x" | "y" | "z" |
    "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" | "I" | "J" | "K" |
    "L" | "M" | "N" | "O" | "P" | "Q" | "R" | "S" | "T" | "U" | "V" |
    "W" | "X" | "Y" | "Z"
数字 (digit) ::= "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" |
    "8" | "9"
分離子 (separator) ::= 空白 | タブ | 改行 | 注釈
注釈 (comment) ::= "{" { 注釈要素 } "}" | "/" "*" { 注釈要素 } "*" "/"
```




入力



入力ファイルは ASCII コードによるテキストファイルであるとしてよい。



あまりにも長い行 (例えば, 1 行 1000 文字以上など) はないものとしてよいが, あったとしても, 作成したプログラムが実行時エラーを起こしてはいけない。

```
プログラム (program) ::= { 字句 | 分離子 }
字句 (token) ::= 名前 | キーワード | 符号なし整数 | 文字列 | 記号
名前 (name) ::= 英字 { 英字 | 数字 }
キーワード (keyword) ::= "p" "r" "o" "g" "r" "a" "m" | "v" "a" "r" |
    "a" "r" "r" "a" "y" | "o" "f" | "b" "e" "g" "i" "n" | "e" "n" "d" |
    "i" "f" | "t" "h" "e" "n" | "e" "l" "s" "e" |
    "p" "r" "o" "c" "e" "d" "u" "r" "e" | "r" "e" "t" "u" "r" "n" |
    "c" "a" "l" "l" | "w" "h" "i" "l" "e" | "d" "o" | "n" "o" "t" |
    "o" "r" | "d" "i" "v" | "a" "n" "d" | "c" "h" "a" "r" |
    "i" "n" "t" "e" "g" "e" "r" | "b" "o" "o" "l" "e" "a" "n" |
    "r" "e" "a" "d" | "w" "r" "i" "t" "e" | "r" "e" "a" "d" "l" "n" |
    "w" "r" "i" "t" "e" "l" "n" | "t" "r" "u" "e" |
    "f" "a" "l" "s" "e" | "b" "r" "e" "a" "k"
符号なし整数 (unsigned integer) ::= 数字 { 数字 }
文字列 (string) ::= "'" { 文字列要素 | "'" "''" } "'"
    # "'"はアポストロフィ (シングルクォート) である
記号 (symbol) ::= "+" | "-" | "*" | "=" | "<" ">" | "<" | "<" "=" |
    ">" | ">" "=" | "(" | ")" | "[" | "]" | ":" "=" | "." | "," |
    ":" | ";"
英字 (alphabet) ::= "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" |
    "i" | "j" | "k" | "l" | "m" | "n" | "o" | "p" | "q" | "r" | "s" |
    "t" | "u" | "v" | "w" | "x" | "y" | "z" |
    "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" | "I" | "J" | "K" |
    "L" | "M" | "N" | "O" | "P" | "Q" | "R" | "S" | "T" | "U" | "V" |
    "W" | "X" | "Y" | "Z"
数字 (digit) ::= "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" |
    "8" | "9"
分離子 (separator) ::= 空白 | タブ | 改行 | 注釈
注釈 (comment) ::= "{" { 注釈要素 } "}" | "/" "*" { 注釈要素 } "*" "/"
```





出力

 字句とその出現数の表.

 標準出力へ出力する.


 出力の仕様は以下の通りである.

 表の1行の行頭に字句をダブルクォーテーションで囲んで表示する. 字句には余分なスペースが含まれていても良い.


 続けて, 1つ以上の空白文字 (空白, タブ)を表示し, 字句の個数を表示する.

 1行には1つの字句についての情報のみを表示する.

 名前 (NAME), 文字列 (STRING), 符号なし整数 (NUMBER) はその実体が異なってもそれぞれを同じ字句として扱う.

 出現しない字句については出力しない (出力中に個数0の行はない).

 以上の仕様に基づくと, 出力は右のようになる.

 なお, 分離子は字句ではないので, 注釈などについては出力する必要はない.

```
"(" 8
")" 8
"*" 1
"+" 1
"," 3
"-" 1
"." 1
":" 6
":=" 3
";" 17
">" 1
"NAME" 27
"NUMBER" 4
"STRING" 2
"begin" 5
"call" 3
"do" 1
"end" 5
"integer" 6
"procedure" 3
"program" 1
"readln" 2
"var" 4
"while" 1
"writeln" 2
```




出力



字句や分離子を構成しない文字が現れたとき(コンパイラとしてはエラーである)









その旨(エラーメッセージ)を標準エラー出力(stderr)へ出力し, ファイルの先頭からそこまでの部分について, 出現数の表を標準出力(stdout)へ出力せよ.

```
"(" 8
")" 8
"*" 1
"+" 1
"," 3
"_" 1
"." 1
":" 6
":=" 3
";" 17
">" 1
"NAME" 27
"NUMBER" 4
"STRING" 2
"begin" 5
"call" 3
"do" 1
"end" 5
"integer" 6
"procedure" 3
"program" 1
"readln" 2
"var" 4
"while" 1
"writeln" 2
```






プログラム作成条件

-  入力から字句を切り出す字句解析系 (スキャナ)
-  字句を数え, 表を作成する主手続き
-  以上2つに分割してプログラムを作成せよ.
-  字句解析系は後の課題で再利用するため, 以下のようなモジュール仕様とせよ.
-  必要に応じて, 別途関数を用意してもよい.
-  エラーメッセージは全て**標準エラー出力stderr**に出力せよ.



初期化関数



```
int init_scan(char *filename)
```

-  filename が表すファイルを入力ファイルとしてオープンする.
-  戻り値:
 -  正常な場合0, ファイルがオープンできない場合など異常な場合は負の値



トークンを1つスキャンする関数

```
int scan()
```

-  次のトークンのコードを返す.
-  トークンコードは別ファイル (scan.h) 参照のこと. End-of-File 等次のトークンをスキャンできないとき, 戻り値として負の値を返す.



定数属性

```
int num_attr;
```







scan() の戻り値が「符号なし整数」のとき、その値を格納している。
なお、32768 よりも大きい値の場合は、エラーである。



文字列属性




```
char string_attr[MAXSTRSIZE];
```

-  scan() の戻り値が「名前」または「文字列」のとき、その実際の文字列を格納している。また、それが「符号なし整数」のときは、入力された数字列を格納している。
-  その文字列 (数字列, 名前) は, '\0' で終端されている。
-  文字列が 'lt's' のときには, string_attrには先頭から順に, 'l', 't', '\', '\', 's', '\0' が格納される。なお, 文字列の定義でも述べたとおり, この文字列の長さは 4 とする (2 つの ' を 1 つと数える)。
-  もし, string_attr に格納できないくらい長い文字列 (数字列, 名前) の場合には, エラーとする。



行番号関数

```
int get_linenum()
```

-  もっとも最近にscan() で返されたトークンが存在した行の番号を返す.
-  まだ一度も scan() が呼ばれていないときには 0 を返す.
-  正確に行番号を数えるのは意外に難しい.



終了処理関数

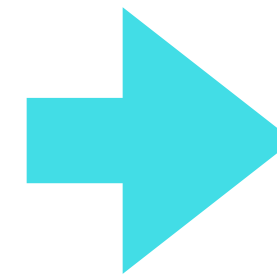
```
void end_scan()
```



init_scan()でオープンしたファイルをクローズする.


```
program sample11pp;
procedure kazuyomikomi(n : integer);
begin
    writeln('input the number of data');
    readln(n)
end;
var sum : integer;
procedure wakakidasi;
begin
    writeln('Sum of data = ', sum)
end;
var data : integer;
procedure goukei(n, s : integer);
    var data : integer;
begin
    s := 0;
    while n > 0 do begin
        readln(data);
        s := s + data;
        n := n - 1
    end
end;
var n : integer;
begin
    call kazuyomikomi(n);
    call goukei(n * 2, sum);
    call wakakidasi
end.
```




入出力例



"NAME	"	27
"program	"	1
"var	"	4
"begin	"	5
"end	"	5
"procedure	"	3
"call	"	3
"while	"	1
"do	"	1
"integer	"	6
"readln	"	2
"writeln	"	2
"NUMBER	"	4
"STRING	"	2
"+	"	1
"-	"	1
"*	"	1
">	"	1
"("	"	8
")"	"	8
" :=	"	3
" .	"	1
" ,	"	3
" :	"	6
" ;	"	17

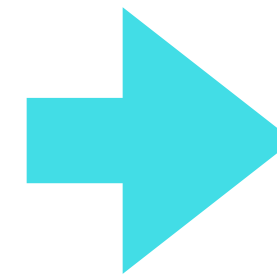


拡張仕様

-  名前についてはその実体ごとにも出現個数を数えて出力するように拡張してみよ.
-  つまり, `n`, `sum` 等の名前毎にも出現個数を数えて出力する.
-  出力の際は, 行頭に "Identifier" を付して名前を続けること.



```
program sample11pp;
procedure kazuyomikomi(n : integer);
begin
    writeln('input the number of data');
    readln(n)
end;
var sum : integer;
procedure wakakidasi;
begin
    writeln('Sum of data = ', sum)
end;
var data : integer;
procedure goukei(n, s : integer);
    var data : integer;
begin
    s := 0;
    while n > 0 do begin
        readln(data);
        s := s + data;
        n := n - 1
    end
end;
var n : integer;
begin
    call kazuyomikomi(n);
    call goukei(n * 2, sum);
    call wakakidasi
end.
```

入出力例




```
"NAME" 27
  "Identifier" "s" 4
  "Identifier" "goukei" 2
  "Identifier" "data" 4
  "Identifier" "wakakidasi" 2
  "Identifier" "sum" 3
  "Identifier" "n" 9
  "Identifier" "kazuyomikomi" 2
  "Identifier" "sample11pp" 1
"program" 1
"var" 4
"begin" 5
"end" 5
"procedure" 3
"call" 3
"while" 1
"do" 1
"integer" 6
"readln" 2
"writeln" 2
"NUMBER" 4
"STRING" 2
"+" 1
"-" 1
"*" 1
">" 1
"(" 8
")" 8
":=" 3
"." 1
"," 3
":" 6
";" 17
```



 テキスト 2.6節, 2.7節 に掲載したプログラムはMoodle上にも置いてある.

 main.c, scan.c → 通常課題

 id-list.c → 拡張課題