Esame di Linguaggi e Traduttori

Grammatica di un sottolinguaggio del linguaggio C

Filippo Landi, matr.121120

Obiettivo iniziale:
Grammatica che riconosca un sottolinguaggio del linguaggio C, in particolare: blocchi, dichiarazioni di variabili, assegnamenti, invocazioni di funzioni.

Dettagli implementativi e limitazioni:

- Mi sono limitato a considerare solo il tipo di dato int e le operazioni di somma (+), sottrazione (-), divisione (/) e moltiplicazione (*). Non considero il modulo (% credo funzionerebbe come la divisione a livello di grammatica) e non considero le operazioni di incremento e decremento (++ e - -). Anche l'assegnamento (=) è un assegnamento semplice non ho considerato quelli composti.

- Per il riconoscimento del tipo di dato ho scritto del codice non riportato nella grammatica. Il suo funzionamento consiste nel dividere la stringa al primo spazio dopo una parola, vedere se essa è 'int' e in caso lo sia o no si continua lo stesso.
  Come commentato nel codice in tale porzione uso delle funzioni di libreria per semplificarmi la vita, nel resto del codice non vengono usate funzioni di libreria diverse da quelle già presenti nel preludio.
  In caso non si voglia tale comportamento basta sostituire la funzione 'ident' con la funzione/ produzione 'identificatore'.

- Lo scopo (funzione codice) per come lo ho descritto mi ha portato ad implementare un PDA che lo rappresenta un po' diverso dal resto del codice: so che tutto il codice è un PDA, ma lo scopo è abbastanza particolare poiché manipola una stringa come se fosse uno stack. Magari ci sono soluzioni migliori ma non mi sono venute in mente.
  Per il resto invece si usa l'analisi ricorsiva discendente classica, la grammatica è LL1 (si capisce sempre che produzione seguire guardando il carattere successivo).

- La grammatica riconosce gli spazi ma non gli "a capo".

- Notare che in caso di riga non accettata a terminale si riceve una eccezione dovuta al fatto che il parser arrivato ad un certo punto non trova la regola da applicare, il che mi pare giusto in quanto la stessa grammatica arriverebbe a un punto morto.
  Ciò comunque non limita un eventuale debugging perché l'eccezione dice chiaramente a quale funzione (quindi produzione) il codice si sia fermato.
  Unico caso differente è l'eventualità di arrivare a fine stringa senza aver bilanciato correttamente le graffe e in tal caso si riceve "(False, "")" come errore.

Specificate le limitazioni procedo nello spiegare come utilizzare il codice:
Aprire un terminale, spostarsi alla cartella col file, aprire ghci caricando il file HParser.hs, chiamare la funzione 'parseString' e scrivere del codice C all'interno di doppie virgolette (rispettando le limitazioni del sottolinguaggio se si vuole riconoscere come giusto tale codice).

Grammatica – Filippo Landi 121120

G = < VT , VN , P , S >
dove:

VT = {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 , int , null , { , } , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 0 , + , - , * , / , (,) , = , ; , , , _}

Notare bene: '_' indica il carattere "spazio".

VN = {<codice>,<istruzione>,<identificatore>,<lettera>,<lettera-cifra>,<cifra>,<numero>,<fineistruzione-dichiarazione-assegnamento>,<espressione-multiassegnamento>,<espressione>,<parametrifunzione>,<espressionep-assegnamentop>,<espressionep>,<assegnamento-chiamatafunzione-postespressione>,<assegnamentop-chiamatafunzionep-postespressionep>,<postespressione>,<postespressionep>,<chiamatafunzione-postespressione>,<chiamatafunzionep-postespressionep>, <segno> ,<segnop>}

P = {
<codice> ::= {<codice>}<codice> | <istruzione><codice> | ;<codice> | null

<istruzione> ::= <identificatore><fineistruzione-dichiarazione-assegnamento>

<identificatore> ::= <lettera><lettera-cifra> | <lettera>

<lettera> ::= 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

<lettera-cifra> ::= <lettera><lettera-cifra> | <cifra><lettera-cifra> | <lettera> | <cifra>

<cifra> ::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0

<numero> ::= <cifra> | <cifra><numero>

<fineistruzione-dichiarazione-assegnamento> ::= ; | ,<istruzione> | =<espressione-multiassegnamento> | _<fineistruzione-dichiarazione-assegnamento>

<espressione-multiassegnamento> ::= (+ | -)<espressione> | <numero><postespressione> | <identificatore><assegnamento-chiamatafunzione-postespressione> | (<espressionep-assegnamentop><postespressione> | _<espressione-multiassegnamento>

<espressione> ::= <identificatore><chiamatafunzione-postespressione> | <numero><postespressione> | (<espressionep-assegnamentop><postespressione> | _<espressione>

<parametrifunzione> ::=) | <espressionep-assegnamentop> | _<parametrifunzione>

<espressionep-assegnamentop> ::= (+ | -)<espressionep> | <numero><postespressionep>| <identificatore><assegnamentop-chiamatafunzionep-postespressionep>| (<espressionep-assegnamentop><postespressionep>| _<espressionep-assegnamentop>

<espressionep> ::= <identificatore><chiamatafunzionep-postespressionep> |
<numero><postespressionep>| (<espressionep-assegnamentop><postespressionep>|
_<espressionep>

<assegnamento-chiamatafunzione-postespressione> ::= =<espressione-multiassegnamento> |
<chiamatafunzione-postespressione> | _<assegnamento-chiamatafunzione-postespressione>

<assegnamentop-chiamatafunzionep-postespressionep> ::= =<espressionep-assegnamentop> |
<chiamatafunzionep-postespressionep> | _<assegnamentop-chiamatafunzionep-postespressionep>

<postespressione> ::= (+ | - | * | /)<segno> | ; | ,<istruzione> | _<postespressione>

<postespressionep> ::= (+ | - | * | /)<segnop> |) | ,<espressionep-assegnamentop> |
_<postespressione>

<chiamatafunzione-postespressione> ::= (<parametrifunzione><postespressione> |
<postespressione> | _<chiamatafunzione-postespressione>

<chiamatafunzionep-postespressionep> ::= (<parametrifunzione><postespressionep>|
<postespressionep>| _<chiamatafunzionep-postespressionep>

<segno> ::= (+ | -)<espressione> | <espressione> | _<segno>

<segnop> ::= (+ | -)<espressionep> | <espressionep> | _<segnop>

}