Relatório de Desenvolvimento

Universidade do Estado de Santa Catarina UDESC

Bacharelado de Ciência da Computação BCC

Compiladores COM0002

R. Paulo Malschitzki, 200 - Zona Industrial Norte, Joinville - SC, 89219-710

28/09/2022

Arthur Henrique Cavalcanti

Introdução.

Nesse relatório vamos abordar a implementação da primeira parte do trabalho semestral da disciplina Compiladores, o trabalho é para ser desenvolvido na linguagem C/LEX tendo para essa primeira entrega os seguintes requisitos: Definição da gramática (Forma Backus-Naur), especificação FLEX, o próprio relatório, destaques para a compilação/ambiente/linguagem, e por fim análise de pelo menos 2 exemplos.

Desenvolvimento.

A definição da gramática desse trabalho acontece entre as linhas 34 e 97, identificando os tokens dentro de suas regras e os adicionando dentro da estrutura de dados escolhida, no caso uma lista duplamente encadeada. A escolha para uma lista duplamente encadeada foi tomada através de perguntas feitas a colegas a qual já cursaram a disciplina anteriormente e recomendaram uma a implementação de uma lista duplamente encadeada. Após a identificação e inserção dos tokens dentro da lista é criada as funções a quais vão ser usadas em conjunto com a lista.

```
131  Mapa add_to_end_table(Mapa **table, char* token){
132    Mapa *aux, *newTable = malloc(sizeof(Mapa));
133    if(newTable){
134      newTable->token = token;
135      newTable->next = NULL;
136
137      if(*table == NULL){ // is the first?
138        *talbe = newTable;
139        newTable->prev = NULL;
140      }else{
141        aux = *table;
142        while(aux->next){
143          aux = aux->next;
144        }
145        aux->next = newTable;
146        newTable->prev = aux;
147      }
148    }else{
149      printf("ERRO AO ALOCAR MEMORIA\n");
150    }
151
152    return table;
153  }
```

Essa a função ao qual adiciona a entrada dos ---- na lista, (mal entendido na hora da implementação o qual achava que token se tratava da entrada dada pelo usuário e não sobre a identificação dessa entrada como token ID, LOWERCASE, UPPERCASE…)

```
166  void print_table(Mapa no){
167      printf("\n***TABLE***\n");
168      while(no){
169        printf("%d ", no->valor);
170        no = no->next;
171      }
172      printf("\n");
173    }
```

Essa é apenas uma simples função onde irá imprimir todos os --- que foram adicionados a listra.

A outras funções que foram implementadas acreditando se ter um uso para implementações futuras e não são usadas nessa primeira implementação, sendo assim não necessária uma apresentação profunda sobre elas, sendo elas: search_table, add_to_start_table, e transforma_romano que foi implementada para o uso de uma atividade complementar e ainda não está concluída mas se nota a necessidade de pilhas para conseguir cumprir as regras de como números romanos são escritos como VIII, onde só se pode ter no máximo três I's após outro número, mostrando que V(5) agora VIII é 8 V+I+I+I.

Conclusão.

Ainda não se consegue chegar a uma conclusão clara, tendo em vista que é apenas a primeira implementação, mas observações em que durante a implementação tem que se escolher em qual é o melhor passo para se tomar no momento e qual é o melhor passo para se ter em futuras implementações, porém também ver o tempo para cada implementação, e então se decidir, pois uma implementação que pense nas próximas pode custar um tempo ao qual está acima do prazo, por outro lado uma implementação que se preocupa com os problemas de agora está dentro do prazo e talvez podendo ter espaço para otimização.