Relatório - Projeto Compiladores

Gramática

Chavetas {} - 0 ou mais repetições
Parêntesis retos [] - opcional (0 ou 1)

Caso geral:

De uma maneira geral, nos casos onde estão presentes chavetas é aplicada recursividade, ou seja, tem-se um produção onde existe uma nova chamada (à esquerda) da mesma produção e outra onde esta chamada não acontece, sendo assim a produção terminal.

Em relação aos parêntesis retos, criamos uma produção onde o seu conteúdo está presente e outra onde ele não está, tornando-o opcional.

Observações:

O símbolo não terminal “CommaId” foi tratado com recursividade à direita. Surge como auxiliar ao símbolo “VarSpec” e facilita o ciclo de declarações de variáveis do mesmo tipo. Como o tipo das variáveis aparece por último, dá-se um tipo fictício aos nós das variáveis (“loopVarSpec”) e, quando se chega ao tipo, cria-se o nó e associa-se o seu “type” ao campo “type” de todas as variáveis.

O símbolo não terminal “ParametersAux” também foi trabalhado com recursividade à direita. Surge como auxiliar ao símbolo “Parameters” e facilita a enumeração dos parâmetros com vírgulas de separação.

Estrutura de Dados

NOTA: Algumas das variáveis descritas abaixo podem não estar presentes na última submissão no Mooshak. Foram adicionadas algumas durante o tratamento de erros semânticos, mas uma vez que o código obteve menos pontuação do que o submitido anteriormente (com a tabela e a árvore anotada 100% funcionais), optou-se por voltar a submeter o código com maior pontuação.

Submissão com 167 pontos - Meta3 (100% nos problemas B,C e D): #2439
Submissão com 82 pontos - Meta3 (tentativa de tratamento de erros): #2438

Árvore de Sintaxe Abstrata (e Árvore de Sintaxe Abstrata Anotada)

A árvore de sintaxe abstrata (AST) e a AST anotada são a mesma estrutura. A construção da AST acontece à medida que são feitos os reduces na gramática. Esta árvore é percorrida inicialmente para a construção da tabela de símbolos, e uma segunda vez, onde acontece a sua anotação. Cada um dos nós da árvore é representado por elementos do tipo “struct node”, descrito abaixo.

Os dados foram guardados em nós contendo o token referente, o seu type, um ponteiro para o filho e outro para o irmão. Por sua vez, estes nós são guardados numa árvore de sintaxe abstrata.

struct node

- **char* type** e **char* token** - tipo (ex.: ID, IntLit, Call, VarDecl, ...) e valor (ex.: nome variável, nome função, valor, …) do nó da árvore.
- **struct node* child** - filho direto do nó.
- **struct node* brother** - irmão do nó.
- **char* note** - anotação do nó (é NULL para todos os nós da árvore enquanto não for feita a análise semântica).
- **int line** e **int column** - linha e coluna para informação de erros semânticos.

Tabela de símbolos

Para a tabela de símbolos, foram criadas 3 structs, uma principal (nodeTab) e duas auxiliares (param e var). A principal representa cada um dos nós (funções ou variáveis globais) da tabela de símbolos (lista ligada). No caso de o nó da tabela representar uma função, as structs auxiliares representam a lista dos seus parâmetros e das variáveis declaradas no seu interior.

struct nodeTab

- **char *label** - categoria do nó, “VarDecl” (variável), “FuncDecl” (função) ou “Global” (quando o token é “program”, cabeçalho da tabela).
- **char *varName** e **char *varType** - nome e tipo da variável, são NULL se o nó representar uma função.
- **char *funcName** e **char *returnType** - nome e tipo de return da função (“none” se a função não retornar nada).
- **struct param* rootParams** e **struct var* rootVars** - se o nó representar uma função, são os primeiros elementos da lista dos seus parâmetros e das variáveis declaradas no seu interior, respetivamente.
- **bool anoted** - indica se a função já foi anotada, serve para apanhar o erro relacionados com redeclarações.
- **struct nodeTab* next** - ponteiro para o nó seguinte (lista ligada).

struct param

- **char *paramName** e **char *paramType** - nome e tipo do parâmetro.
- **struct param* nextParam** - ponteiro para o nó seguinte (lista ligada).

struct var

- **char *varName** e **char *varType** - nome e tipo da variável.
- **struct var* nextVar** - ponteiro para o nó seguinte (lista ligada).
- **int line** - linha em que a variável foi declarada (dentro da função).