README.md

Trab_CC_2022_1

CP1 - Analizadores léxico e sintático

Desenvolvidos a partir da gramática Yacc baseada no C99 disponibilizada aqui.

Modificações / Simplificações

- Correção do conflito *shift-reduce* provocado pelo *else* pendente (*else* agora é associado ao *if* mais próximo);
- Inclusão de mensagens de erro exibindo linha e coluna;
- Inclusão do *carriage-return* aos *whitespaces*;
- Simplificações: Remoção de *typedef, struct, union* e *enum*; de todos os tipos exceto: *char, int, float* e *void*; de prefixos e sufixos em literais; de *for loops*; de incremento, decremento e operadores de atribuição; de inicialização de *arrays* (inclusive de *chars*); e de vários aspectos da linguagem fora do escopo do projeto;
- Entrada e saída realizadas com funções *get/put* polimórficas:
  - *put* adiciona *newline*;
  - !!! `put(get());` Não funciona !!!;
  - *put* aceita *string literals*, embora estas não sirvam para inicializar *char arrays*.

Casos de teste

Todos os aspectos da linguagem que serão implementados foram inclusos nos casos de teste, assim comos alguns erros.

TODO

- Mais simplificações?

CP2 - Analizador Semântico e AST

- Tabelas de variáveis, funções e strings implementadas usando *rbt*;
- Checagem de redeclaração de variáveis, considerando escopo, e chamada de variáveis não declaradas;
  - Escopo implementado usando "pilha";
- Checagem de tipo, aceitando apenas a coerção:
  - int -> float;

    - char -> int;
    - char -> float;
- AST dos labs adaptadas para linguagem C e novas tabelas de símbolos.

TODO

- Provavelmente muitos bugs ocultos.....
- Implementar AST de forma não miserável.

Geração de Código

- Linguagem alvo: JVM
- Alguns erros relacionados a *arrays* são detectados apenas ao rodar o *bytecode*, como acesso a um index fora da faixa, uso de *float* como index e tentativa de printar *arrays* que não são do tipo *char*;
- Como método *main* deve retornar *void* no java, programa nunca retorna nada.

Debugging / Scripts

Programa lê duas variáveis de ambiente:

- Caso `CC_DOT` exista e seja igual a 1:
    - O programa imprimirá a árvore do programa lido;
- Caso `CC_ST` exista e seja igual a 1:
    - O programa imprimirá as tabelas de símbolos.

Scripts

- `test`:
    - Se `CC_DOT=1`, criará uma pasta `./trees` e gerará os pdfs de cada caso de teste;
    - Senão criará uma pasta `./out` e gerará os arquivos *bytecode*.
- `run`: Compila e roda um programa. Deleta os arquivos `.j` e `.class` em seguida;
- `runall`: Executa o `run` para todos os casos de teste (! Alguns casos devem esperar input !);

TODO

- Provavelmente mais bugs ocultos.....
- Encontrar forma mais inteligente de construir a AST/gerar o código.