A Linguagem de Programação C-

Tem especificação similar à linguagem C, porém simplificada. Admite inteiros, vetores de inteiros, funções e recursão.

Convenções Léxicas:

Palavras-chave: else, if, int, return, void, while
Símbolos especiais: + - * / < <= > >= == != = ; , { } [ ] /* */
Identificadores: nome inicia por letra, seguido de 0 ou mais letras
    ID = letra letra*
    letra = a|..|z|A|..|Z
Números: inteiros sem sinal
    NUM = digito digito*
    digito = 0|..|9
Espaço em branco: brancos, mudança de linha e tabulação
Comentários: entre /* e */ (não podem ser aninhados)

Gramática BNF para C-:

programa → declaração-lista
declaração-lista → declaração-lista declaração | declaração
declaração → var-declaração | fun-declaração
var-declaração → tipo-especificador ID ; | tipo-especificador ID [NUM] ;
tipo-especificador → int | void
fun-declaração → tipo-especificador ID (params) composto-decl
params → param-lista | void
param-lista → param-lista, param | param
param → tipo-especificador ID | tipo-especificador ID [ ]
composto-decl → { local-declarações statement-lista } | {local-declarações} | {statement-lista} | { }
local-declarações → local-declarações var-declaração | var-declaração
statement-lista → statement-lista statement | statement
statement → expressão-decl | composto-decl | seleção-decl | iteração-decl | retorno-decl
expressão-decl → expressão ; | ;
seleção-decl → if (expressão) statement | if (expressão) statement else statement
iteração-decl → while (expressão) statement
retorno-decl → return ; | return expressão;
expressão → var = expressão | simples-expressão
var → ID | ID [expressão]
simples-expressão → soma-expressão relacional soma-expressão | soma-expressão
relacional → <= | < | > | >= | == | !=
soma-expressão → soma-expressão soma termo | termo
soma → + | -
termo → termo mult fator | fator
mult → * | /
fator → (expressão) | var | ativação | NUM
ativação → ID (arg-lista) | ID ( )
arg-lista → arg-lista , expressão | expressão

Restrições semânticas:

- Todas as variáveis e funções devem ser declaradas antes do uso.
- A última declaração deve ser de função, da forma void main(void).

- Void só é usado em declarações de função.
- Apenas uma variável pode ser declarada em cada declaração.
- Um retorno transfere o controle de volta para o ativador (ou termina o programa se ocorrer dentro de main)
- Um índice negativo em vetor leva à interrupção do programa.
- Os limites superiores dos índices não são verificados.
- Não permite aritmética de ponteiros.
- A associatividade e a precedência dos operadores é a padrão em aritmética.
- A divisão é inteira, ou seja, o resto é truncado.
- A quantidade de parâmetros na ativação de uma função deve ser igual à contida em sua declaração e deve seguir o tipo declarado.
- A entrada de valores (inteiros) é feita por uma função:
  int input(void) { … }
  que pressupõe-se que seja pré-definida no ambiente global (nativa da linguagem).
- A saída de valores ocorre pela função:
  void output(int x) { … }
  que também pressupõe-se pré-definida.

  Verificações semânticas necessárias:
  - Verificar se variável foi declarada antes de uso;
  - Verificação de tipos (variável inteira não pode receber retorno de função void);
  - Não pode ter variáveis void;
  - Não podem haver declarações duplas no mesmo escopo;
  - Função tem que ter sido definida antes de uso;
  - Todo programa tem que ter main;
  - Nomes de variáveis e funções não podem ser duplicados.