PCS2056
Linguagens e Compiladores
Definição da linguagem

Professor: Ricardo Luis de Azevedo Rocha

Grupo:
Filipe Morgado Simões de Campos 5694101
Rafael Barbolo Lopes 5691262

Gramática em notação BNF

```
<programa> ::= <funções> main { <declarações> begin <comandos>}

<declarações> ::= <tipo> <variável>; | <tipo> <variável>, <declarações> | ε
<variável> ::= <id> | <vetor> | <matriz>
<vetor> ::= <id>[<inteiro>]
<matriz> ::= <id>[<inteiro>][<inteiro>]

<funções> ::= <função> | <função> <funções> | ε
<função> ::= <tipo_retorno> <id>(argumentos) {<declarações> begin <comandos>
            <retorno>}
<tipo_retorno> ::= <tipo> | void
<argumentos> ::= <tipo> <variável> | <tipo> <variável>, <argumentos> | ε
<retorno> ::= return <expressão>; | ε
<chamada_função> ::= <id>(argumentos)

<comandos> ::= <comando> | <comando> <comandos> | ε
<comando> ::= <atribuição> | <condicional> | <iteração> | <entrada> |
            <saída> | <chamada_função>;
<atribuição> ::= <variável> = <expressão>;
<condicional> ::= if (<condição>) {<comandos>} |
                  if (<condição>) {<comandos>} else {<comandos>}
<iteração> ::= while (condição) {<comandos>}
<entrada> ::= scanf(<captura_entrada>);
<captura_entrada> ::= "<tipo_entrada>", <id> | "<tipo_entrada", <id>,
            <captura_entrada>
<tipo_entrada> ::= %c | %d | %f | %b
<saída> ::= printf(<texto>);

<id> ::= <letra><letras_ou_dígitos>
<tipo> ::= int | float | char | boolean

<expressão> ::= <expressão> + <termo> | <expressão> - <termo> |
            pow(<expressão>,<expressão>) | <termo>
<termo> ::= <termo> * <fator> | <termo> / <fator> | <fator>
<fator> ::= (<expressão>) | <sinal><valor>

<condição> ::= <expressão> <operador_booleano> <expressão> |
            <expressão_booleana>
<operador_booleano> ::= < | <= | > | >= | == | !=
<expressão_booleana> ::= <condição> <operador_lógico> <condição> |
            <booleano> | (<condição>) | NOT <expressão_booleana>
<operador_lógico> ::= AND | OR

<texto> ::= "<caracteres>" | "<caracteres>" + <texto> | <conteúdo> |
            <conteúdo> + <texto>
<conteúdo_id> ::= '<expressão>' | '<condição>'

<sinal> ::= - | ε
<valor> ::= <variável> | <número> | <chamada_função> | <booleano>
<número> ::= <inteiro> | <decimal>
<inteiro> ::= <dígito> | <dígito><inteiro>
<decimal> ::= <inteiro>.<inteiro>
<booleano> ::= true | false

<letras_ou_dígitos> ::= <letra_ou_dígito> | <letra_ou_dígito>
            <letras_ou_dígitos>
<letra_ou_dígito> ::= <letra> | <dígito>
<letra> ::=  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
<dígito> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<caracteres> ::= <letra> | <digito> | + | - | / | * | =
```

Gramática em notação de Wirth

```
programa = funções “main” “{” declarações “begin” comandos “}”.

declarações = [tipo variável {“,” tipo variável}“;”].
variável = id | vetor | matriz.
vetor = id”[“inteiro”]”.
matriz = id”[“inteiro”]””[“inteiro”]”.

funções = {função}.
função = tipo_retorno id “(“ argumentos “)” “{“ declarações “begin” comandos
          retorno “}”.
tipo_retorno = tipo | “void”.
argumentos = [tipo variável {“,” tipo variável }].
retorno = [“return” expressão “;”].
chamada_função = id “(“ argumentos “)”.

comandos = {comando}.
comando = atribuição | condicional | iteração | entrada | saída |
          chamada_função.
atribuição = variável “=” expressão “;”.
condicional = “if” “(“ condição “)” “{“ comandos “}” [“else” “{“ comandos
          “}”].
iteração = “while” “(“ condição “)” “{“ comandos “}”.
entrada = “scanf” “(“ captura_entrada “)”.
captura_entrada = “”” tipo_entrada “””, id {,””” tipo_entrada “””, id}.
tipo_entrada = “%c” | “%d” | “%f” | “%b”.
saída = “printf” “(“ texto “)” “;”.

id = letra{letra | dígito}.
tipo = “int” | “float” | “char” | “boolean”.

expressão = expressão (“+”|”-“) termo | “pow” “(“ expressão, expressão “)” |
          termo.
termo = termo (“*”|”/”) fator.
fator = “(“ expressão “)” | sinal valor.

condição = expressão operador_booleano expressão | expressão_booleana.
operador_booleano = "<" | "<=" | ">" | ">=" | "==" | "!=".
expressão_booleana = condição operador_lógico condição | booleano | “(“
          condição “)” | “NOT” expressão_booleana.
operador_lógico = “AND” | “OR”.

texto = (””” caracteres ””” | conteúdo){“+” (””” caracteres ””” |
          conteúdo)}.
sinal = [“-“].
valor = variável | número | chamada_função | booleano.
número = inteiro | decimal.
inteiro = dígito{dígito}.
decimal = inteiro”.”inteiro.
booleano = “true” | “false”.

letra = "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 ".
dígito = “0” | “1” | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9".
caracteres = {letra | dígito | "+" | "-" | "*" | "/" | "="}.
```

Palavras reservadas

main
begin
void
return
if
else
while
scanf
printf
int
float
char
boolean
pow
not
and
or
true
false

Símbolos compostos de mais de um caracter ascii

%c
%d
%f
%b
<=
>=
==
!=

Exemplo de programa

```
int equacao1(int a, int b) {
        int resultado;

        begin

        resultado = a + b*a;

        return resultado;
}

int equacao2(int a, int b) {
        int resultado;

        begin

        resultado = a - b/a;

        return resultado;
}



main {
        int operacao, a, b;

        begin

        a = 10;
        b = 2;
        while true {
            scanf("%d", operacao);
            if (operacao == 1) {
                printf(equacao1(a,b));
            }
            else {
                if (operacao == 2) {
                    printf(equacao2(a,b));
                }
                else {
                    printf("operação incorreta");
                }
            }

        }

}
```