UNIVERSIDADE FEDERAL DE ALAGOAS

INSTITUTO DE COMPUTAÇÃO

COMPILADORES - 2017.2

Hapais: gramática e analisador sintático

Autor 1
Dayvson SALES

Autor 2
Warley VITAL

May 31, 2018

Chapter 1

Introdução

O analisador sintático a ser utilizado será o Preditivo Tabular. Nos próximos capítulos serão apresentados as gramáticas e a tabela gerada, além das saídas dos exemplos de acordo com a especificação.

Chapter 2

Gramática Livre de Contexto

```
1. S = 'defmod' 'id' 'do' Global DeclFuncoes
   'endmod'
2. Global = DeclVar

3. DeclFuncoes = DeclFuncoes Funcao
4. DeclFuncoes = Funcao
5. Funcao = 'def' TipoFuncao 'id' '('
   Parametro ')' 'do' Instrucao 'end'

6. Return = 'return' Exp ';'

7. TipoFuncao = Tipo '[' ']'
8. TipoFuncao = Tipo

9. Tipo = 'int'
10. Tipo = 'void'
11. Tipo = 'str'
12. Tipo = 'real'
13. Tipo = 'bool'
14. Tipo = 'char'
15. TipoFixo = Tipo
16. TipoFixo = Tipo '[' ']'
17. TipoFixo = Tipo '[' CteInt ']'

18. Parametro = 'id' ':' TipoFuncao
19. Parametro = Parametro ',' 'id' ':'
    TipoFuncao
20. Parametro = epsilon

21. Instrucao = Instrucao Comando
22. Instrucao = Comando
23. Instrucao = epsilon

24. Comando = Rep
25. Comando = DoUntil
26. Comando = Until
27. Comando = When
28. Comando = TipoFixo 'id' AtrOuNao ';'
29. AtrOuNao = '=' Exp
30. AtrOuNao = ''
31. Comando = 'id' RArr AtrOuFunc ';'
32. AtrOuFunc = '=' Exp
33. AtrOuFunc = '(' ParFunc ')'
34. Comando = Return

35. When = 'when' '(' Exp ')' 'do' Instrucao
    Otherwise 'end'
36. Otherwise = 'otherwise' Instrucao
37. Otherwise = ''
38. Until = 'until' '(' Exp ')' 'do'
    Instrucao 'end'
39. DoUntil = 'do' Instrucao 'till' '(' Exp
    ')' 'end'
40. Rep = 'rep' '(' VarControl ',' Exp ',' Id
    '=' Exp ')' 'do' Instrucao 'end'
41. VarControl = 'id' ':' Tipo '=' Exp

42. Atrib = Id '=' Exp ';'

43. Id = 'id' Arr

44. DeclVar = DeclVar LDeclVar
45. DeclVar = LDeclVar
46. LDeclVar = TipoFixo Atrib ';'
47. LDeclVar = TipoFixo ';'
48. DeclVar = epsilon

49. Arr = '[' Exp ']'
50. Arr = epsilon
51. RArr = '[' LRArr ']'
52. RArr = epsilon
53. LRArr = Exp
54. LRArr = epsilon

55. ParFunc = ParFunc ',' Exp
56. ParFunc = Exp
57. ParFunc = epsilon

# ordem:  - ~,  * /,  + -, $, < > <= >=,  ==
    !=, &&, ||

58. Exp = Exp '||' ExpBoolAnd
59. Exp = ExpBoolAnd

60. ExpBoolAnd = ExpBoolAnd '&&' Exprdois
61. ExprBooldAnd = Exprdois

62. ExpRdois = ExpRdois 'opr2' ExpRum
63. ExpRdois = ExpRum

64. ExpRum = ExpRum 'opr1' ExpConcat
65. ExpRum = ExpConcat

66. ExpConcat = ExpConcat '$' Expa
```

```
67. ExpConcat = Expa

68. Expa = Expa 'opa' Expm
69. Expa = Expm

70. Expm = Expm 'opm' Expu
71. Expm = Expu

72. Expu = Unario Fa
73. Expu = Fa

74. Unario = '~'
75. Unario = '-'
```

```
76. Fa = '(' Exp ')'
77. Fa = 'id'
78. Fa = 'id' '(' ParFunc ')'
79. Fa = 'id' RArr
80. Fa = Cte

81. Cte = 'cteI'
82. Cte = 'cteR'
83. Cte = 'cteStr'
84. Cte = 'cteChar'
85. Cte = CteBool
86. CteBool = 'true'
87. CteBool = 'false'
```

Chapter 3

Gramática LL(1)

1. S = 'defmod' 'id' 'do' Global DeclFuncoes 'endmod'
2. Global = RDeclVar
3. DeclFuncoes = Funcao DeclFuncoes
4. DeclFuncoes = epsilon
5. Funcao = 'def' TipoFuncao 'id' '(' Parametro ')' 'do' Instrucao 'end'
6. Return = 'return' Exp ';'
7. TipoFuncao = Tipo FArr
8. FArr = '[' RFArr ']'
9. RFArr = Exp
10. RFArr = epsilon
11. FArr = epsilon
12. Tipo = 'int'
13. Tipo = 'void'
14. Tipo = 'str'
15. Tipo = 'real'
16. Tipo = 'bool'
17. Tipo = 'char'
18. TipoFixo = Tipo FArr
19. Parametro = 'id' ':' TipoFixo Parametro
20. Parametro = ',' 'id' ':' TipoFixo
21. Parametro = epsilon
22. Instrucao = Comando RInstrucao
23. RInstrucao = epsilon
24. Comando = Rep Comando
25. Comando = DoUntil Comando
26. Comando = Until Comando
27. Comando = When Comando
28. Comando = TipoFixo 'id' ComandoA ';' Comando
29. Comando = 'id' RArr ComandoX Comando
30. ComandoX = = Exp ';'
31. ComandoX = '(' ParFunc ')' ';'
32. ComandoA = = Exp
33. ComandoA = epsilon
34. Comando = Return
35. Comando = epsilon
36. Atrib = Id = Exp ';'
37. Atrib = epsilon
38. Id = 'id' Arr
39. RDeclVar = DeclVar RDeclVar
40. RDeclVar = epsilon
41. DeclVar = TipoFixo Atrib ';'
42. Arr = '[' Exp ']'
43. Arr = epsilon
44. RArr = '[' LRArr ']'
45. RArr = epsilon
46. LRArr = Exp
47. LRArr = epsilon
48. When = 'when' '(' Exp ')' 'do' Otherwise 'end'
49. Otherwise = Instrucao OtherwiseR
50. OtherwiseR = 'otherwise' Instrucao
51. OtherwiseR = epsilon
52. Until = 'until' '(' Exp ')' 'do' Instrucao 'end'
53. DoUntil = 'do' Instrucao 'till' '(' Exp ')' 'end'
54. Rep = 'rep' '(' VarControl ',' Exp ',' Id = Exp ')' 'do' Instrucao 'end'
55. VarControl = 'id' ':' Tipo = Exp
56. ParFunc = RParFunc
57. ParFunc = epsilon
58. RParFunc = Exp TRParFunc
59. TRParFunc = ',' Exp TParFunc
60. TRParFunc = epsilon
61. TParFunc = epsilon
62. Exp = ExpBoolAnd RExp
63. RExp = '||' ExpBoolAnd RExp
64. RExp = epsilon
65. ExpBoolAnd = ExpRDois RExpBoolAnd
66. RExpBoolAnd = '&&' ExpRDois RExpBoolAnd
67. RExpBoolAnd = epsilon
68. ExpRDois = ExpRUm RExpRDois
69. RExpRDois = 'opr2' ExpRUm
70. RExpRDois = epsilon
71. ExpRUm = ExpConcat RExpRUm
72. RExpRUm = 'opr1' ExpConcat RExpRUm
73. RExpRUm = epsilon
74. ExpConcat = Expa RExpConcat
75. RExpConcat = '$' Expa RExpConcat
76. RExpConcat = epsilon
77. Expa = Expm RExpa
78. RExpa = 'opa' Expm RExpa
79. RExpa = epsilon
80. Expm = ExpU RExpm
81. RExpm = 'opm' ExpU RExpm
82. RExpm = epsilon
83. ExpU = Unario Fa
84. ExpU = Fa
85. Unario = '~'
86. Unario = '-'
87. Fa = '(' Exp ')'

```
88.  Fa = 'id' RFa
89.  RFa = '(' ParFunc ')'
90.  RFa = RArr
91.  Fa = Cte
92.  Cte = 'cteI'
93.  Cte = 'cteR'
94.  Cte = 'cteStr'
95.  Cte = 'cteChar'
96.  Cte = CteBool
97.  CteBool = 'true'
98.  CteBool = 'false'
```

Chapter 4

Tabela do Analisador Preditivo

Em anexo a este documento está a tabela.

Chapter 5

Saída dos exemplos

5.1 Hello World

```
Start:
   1  defmod AloMundo do
   2    def void main() do
   3            print("Alo mundo!");
   4    end
   5  endmod

   1  defmod AloMundo do

        S = 'defmod' 'id' 'do' Global
            DeclFuncoes 'endmod'
            [001, 001] (0001,     defmod) {
                defmod}
            [001, 008] (0000,         id) {
                AloMundo}
            [001, 017] (0003,         do) {
                do}

   2    def void main() do

        Global = RDeclVar
        RDeclVar = epsilon
        DeclFuncoes = Funcao DeclFuncoes
        Funcao = 'def' TipoFuncao 'id' '('
            Parametro ')' 'do' Instrucao '
            end'
            [002, 003] (0002,        def) {
                def}
        TipoFuncao = Tipo FArr
        Tipo = 'void'
            [002, 007] (0011,       void) {
                void}
        FArr = epsilon
            [002, 012] (0000,         id) {
                main}
            [002, 016] (0027,          () {(}
        Parametro = epsilon
            [002, 017] (0028,          )) {)}
            [002, 019] (0003,         do) {
                do}

   3            print("Alo mundo!");
        Instrucao = Comando RInstrucao
        Comando = 'id' RArr ComandoX
            Comando
            [003, 005] (0025,         id) {
                print}
        RArr = epsilon
        ComandoX = '(' ParFunc ')' ';'
            [003, 010] (0027,          () {(}
        ParFunc = RParFunc
        RParFunc = Exp TRParFunc
        Exp = ExpBoolAnd RExp
        ExpBoolAnd = ExpRDois RExpBoolAnd
        ExpRDois = ExpRUm RExpRDois
        ExpRUm = ExpConcat RExpRUm
        ExpConcat = Expa RExpConcat
        Expa = Expm RExpa
        Expm = ExpU RExpm
        ExpU = Fa
        Fa = Cte
        Cte = 'cteStr'
            [003, 011] (0019,     cteStr) {
                Alo mundo!}
        RExpm = epsilon
        RExpa = epsilon
        RExpConcat = epsilon
        RExpRUm = epsilon
        RExpRDois = epsilon
        RExpBoolAnd = epsilon
        RExp = epsilon
        TRParFunc = epsilon
            [003, 023] (0028,          )) {)}
            [003, 024] (0023,          ;) {;}

   4    end

        Comando = epsilon
        RInstrucao = epsilon
            [004, 003] (0004,        end) {
                end}

   5  endmod
```

```
        DeclFuncoes = epsilon
            [005, 001] (0005,     endmod) {
                endmod}
```

```
Aceito!
End:
```

5.2 Fibonacci

```
Start:
   1  defmod Fibonacci do
   2    def void fib(limite : int) do
   3      int[limite+1] arr;
   4      arr[0] = 0;
   5      arr[1] = 1;
   6      int i = 0;
   7
   8    when(limite <= 0) do
   9            print("0");
  10      end
  11
  12      until(i < limite) do
  13        when (i > 2) do
  14          arr[i] = arr[i-1] + arr[i-2];
  15        end
  16
  17        print(arr[i]);
  18
  19        when(i != limite-1) do
  20          print(", ");
  21        end
  22
  23        i = i + 1;
  24      end
  25    end
  26
  27
  28    def void main() do
  29      int limite = read();
  30      fib(limite);
  31    end
  32
  33  endmod
```

```
   1  defmod Fibonacci do

        S = 'defmod' 'id' 'do' Global
            DeclFuncoes 'endmod'
            [001, 001] (0001,     defmod) {
                defmod}
            [001, 008] (0000,         id) {
                Fibonacci}
            [001, 018] (0003,         do) {
                do}

   2    def void fib(limite : int) do

        Global = RDeclVar
        RDeclVar = epsilon
        DeclFuncoes = Funcao DeclFuncoes
        Funcao = 'def' TipoFuncao 'id' '('
            Parametro ')' 'do' Instrucao '
            end'
            [002, 003] (0002,        def) {
                def}
        TipoFuncao = Tipo FArr
        Tipo = 'void'
            [002, 007] (0011,       void) {
                void}
        FArr = epsilon
            [002, 012] (0000,         id) {
                fib}
            [002, 015] (0027,          () {(}
        Parametro = 'id' ':' TipoFixo
            Parametro
            [002, 016] (0000,         id) {
                limite}
            [002, 023] (0022,          :) {:}
        TipoFixo = Tipo FArr
        Tipo = 'int'
            [002, 025] (0012,        int) {
                int}
        FArr = epsilon
        Parametro = epsilon
            [002, 028] (0028,          )) {)}
            [002, 030] (0003,         do) {
                do}

   3    int[limite+1] arr;

        Instrucao = Comando RInstrucao
        Comando = TipoFixo 'id' ComandoA
            ';' Comando
        TipoFixo = Tipo FArr
        Tipo = 'int'
            [003, 005] (0012,        int) {
                int}
        FArr = '[' RFArr ']'
            [003, 008] (0029,          [] {[}
        RFArr = Exp
        Exp = ExpBoolAnd RExp
        ExpBoolAnd = ExpRDois RExpBoolAnd
        ExpRDois = ExpRUm RExpRDois
        ExpRUm = ExpConcat RExpRUm
        ExpConcat = Expa RExpConcat
        Expa = Expm RExpa
        Expm = ExpU RExpm
        ExpU = Fa
        Fa = 'id' RFa
            [003, 009] (0000,         id) {
                limite}
        RFa = RArr
        RArr = epsilon
        RExpm = epsilon
        RExpa = 'opa' Expm RExpa
            [003, 015] (0034,        opa) {+}
```

```
        Expm = ExpU RExpm
        ExpU = Fa
        Fa = Cte
        Cte = 'cteI'
            [003, 016] (0017,       cteI)
                {1}
        RExpm = epsilon
        RExpa = epsilon
        RExpConcat = epsilon
        RExpRUm = epsilon
        RExpRDois = epsilon
        RExpBoolAnd = epsilon
        RExp = epsilon
            [003, 017] (0030,          ])
                {]}
            [003, 019] (0000,         id) {
                arr}
        ComandoA = epsilon
            [003, 022] (0023,          ;)
                {;}

4       arr[0] = 0;

        Comando = 'id' RArr ComandoX
            Comando
            [004, 005] (0000,         id) {
                arr}
        RArr = '[' LRArr ']'
            [004, 008] (0029,          [)
                {[}
        LRArr = Exp
        Exp = ExpBoolAnd RExp
        ExpBoolAnd = ExpRDois RExpBoolAnd
        ExpRDois = ExpRUm RExpRDois
        ExpRUm = ExpConcat RExpRUm
        ExpConcat = Expa RExpConcat
        Expa = Expm RExpa
        Expm = ExpU RExpm
        ExpU = Fa
        Fa = Cte
        Cte = 'cteI'
            [004, 009] (0017,       cteI)
                {0}
        RExpm = epsilon
        RExpa = epsilon
        RExpConcat = epsilon
        RExpRUm = epsilon
        RExpRDois = epsilon
        RExpBoolAnd = epsilon
        RExp = epsilon
            [004, 010] (0030,          ])
                {]}
        ComandoX = '=' Exp ';'
            [004, 012] (0036,          =)
                {=}
        Exp = ExpBoolAnd RExp
        ExpBoolAnd = ExpRDois RExpBoolAnd
        ExpRDois = ExpRUm RExpRDois
        ExpRUm = ExpConcat RExpRUm
        ExpConcat = Expa RExpConcat
        Expa = Expm RExpa
        Expm = ExpU RExpm
        ExpU = Fa
        Fa = Cte
        Cte = 'cteI'
            [004, 014] (0017,       cteI)
                {0}
        RExpm = epsilon
        RExpa = epsilon
        RExpConcat = epsilon
        RExpRUm = epsilon
        RExpRDois = epsilon
        RExpBoolAnd = epsilon
        RExp = epsilon
            [004, 015] (0023,          ;)
                {;}

5       arr[1] = 1;

        Comando = 'id' RArr ComandoX
            Comando
            [005, 005] (0000,         id) {
                arr}
        RArr = '[' LRArr ']'
            [005, 008] (0029,          [)
                {[}
        LRArr = Exp
        Exp = ExpBoolAnd RExp
        ExpBoolAnd = ExpRDois RExpBoolAnd
        ExpRDois = ExpRUm RExpRDois
        ExpRUm = ExpConcat RExpRUm
        ExpConcat = Expa RExpConcat
        Expa = Expm RExpa
        Expm = ExpU RExpm
        ExpU = Fa
        Fa = Cte
        Cte = 'cteI'
            [005, 009] (0017,       cteI)
                {1}
        RExpm = epsilon
        RExpa = epsilon
        RExpConcat = epsilon
        RExpRUm = epsilon
        RExpRDois = epsilon
        RExpBoolAnd = epsilon
        RExp = epsilon
            [005, 010] (0030,          ])
                {]}
        ComandoX = '=' Exp ';'
            [005, 012] (0036,          =)
                {=}
        Exp = ExpBoolAnd RExp
        ExpBoolAnd = ExpRDois RExpBoolAnd
        ExpRDois = ExpRUm RExpRDois
        ExpRUm = ExpConcat RExpRUm
        ExpConcat = Expa RExpConcat
        Expa = Expm RExpa
        Expm = ExpU RExpm
        ExpU = Fa
        Fa = Cte
        Cte = 'cteI'
            [005, 014] (0017,       cteI)
                {1}
        RExpm = epsilon
        RExpa = epsilon
        RExpConcat = epsilon
```

```
        RExpRUm = epsilon
        RExpRDois = epsilon
        RExpBoolAnd = epsilon
        RExp = epsilon
            [005, 015] (0023,          ;)
                {;}

6       int i = 0;

        Comando = TipoFixo 'id' ComandoA
            ';' Comando
        TipoFixo = Tipo FArr
        Tipo = 'int'
            [006, 005] (0012,        int) {
                int}
        FArr = epsilon
            [006, 009] (0000,         id) {
                i}
        ComandoA = '=' Exp
            [006, 011] (0036,          =)
                {=}
        Exp = ExpBoolAnd RExp
        ExpBoolAnd = ExpRDois RExpBoolAnd
        ExpRDois = ExpRUm RExpRDois
        ExpRUm = ExpConcat RExpRUm
        ExpConcat = Expa RExpConcat
        Expa = Expm RExpa
        Expm = ExpU RExpm
        ExpU = Fa
        Fa = Cte
        Cte = 'cteI'
            [006, 013] (0017,       cteI)
                {0}
        RExpm = epsilon
        RExpa = epsilon
        RExpConcat = epsilon
        RExpRUm = epsilon
        RExpRDois = epsilon
        RExpBoolAnd = epsilon
        RExp = epsilon
            [006, 014] (0023,          ;)
                {;}

8   when(limite <= 0) do

        Comando = When Comando
        When = 'when' '(' Exp ')' 'do'
            Otherwise 'end'
            [008, 002] (0008,       when) {
                when}
            [008, 006] (0027,          ()
                {(}
        Exp = ExpBoolAnd RExp
        ExpBoolAnd = ExpRDois RExpBoolAnd
        ExpRDois = ExpRUm RExpRDois
        ExpRUm = ExpConcat RExpRUm
        ExpConcat = Expa RExpConcat
        Expa = Expm RExpa
        Expm = ExpU RExpm
        ExpU = Fa
        Fa = 'id' RFa
            [008, 007] (0000,         id) {
                limite}
        RFa = RArr
        RArr = epsilon
        RExpm = epsilon
        RExpa = epsilon
        RExpConcat = epsilon
        RExpRUm = 'opr1' ExpConcat RExpRUm
            [008, 013] (0037,       opr1)
                {<=}
        ExpConcat = Expa RExpConcat
        Expa = Expm RExpa
        Expm = ExpU RExpm
        ExpU = Fa
        Fa = Cte
        Cte = 'cteI'
            [008, 016] (0017,       cteI)
                {0}
        RExpm = epsilon
        RExpa = epsilon
        RExpConcat = epsilon
        RExpRUm = epsilon
        RExpRDois = epsilon
        RExpBoolAnd = epsilon
        RExp = epsilon
            [008, 017] (0028,          ))
                {)}
            [008, 019] (0003,         do) {
                do}

9           print("0");

        Otherwise = Instrucao OtherwiseR
        Instrucao = Comando RInstrucao
        Comando = 'id' RArr ComandoX
            Comando
            [009, 006] (0025,         id) {
                print}
        RArr = epsilon
        ComandoX = '(' ParFunc ')' ';'
            [009, 011] (0027,          ()
                {(}
        ParFunc = RParFunc
        RParFunc = Exp TRParFunc
        Exp = ExpBoolAnd RExp
        ExpBoolAnd = ExpRDois RExpBoolAnd
        ExpRDois = ExpRUm RExpRDois
        ExpRUm = ExpConcat RExpRUm
        ExpConcat = Expa RExpConcat
        Expa = Expm RExpa
        Expm = ExpU RExpm
        ExpU = Fa
        Fa = Cte
        Cte = 'cteStr'
            [009, 012] (0019,     cteStr)
                {0}
        RExpm = epsilon
        RExpa = epsilon
        RExpConcat = epsilon
        RExpRUm = epsilon
        RExpRDois = epsilon
        RExpBoolAnd = epsilon
        RExp = epsilon
        TRParFunc = epsilon
```

```
        [009, 015] (0028,          ))
            {)}
        [009, 016] (0023,          ;)
            {;}

10      end

        Comando = epsilon
        RInstrucao = epsilon
        OtherwiseR = epsilon
        [010, 005] (0004,        end) {
            end}

12      until(i < limite) do

        Comando = Until Comando
        Until = 'until' '(' Exp ')' 'do'
            Instrucao 'end'
        [012, 005] (0006,      until) {
            until}
        [012, 010] (0027,          ()
            {(}
        Exp = ExpBoolAnd RExp
        ExpBoolAnd = ExpRDois RExpBoolAnd
        ExpRDois = ExpRUm RExpRDois
        ExpRUm = ExpConcat RExpRUm
        ExpConcat = Expa RExpConcat
        Expa = Expm RExpa
        Expm = ExpU RExpm
        ExpU = Fa
        Fa = 'id' RFa
        [012, 011] (0000,         id) {
            i}
        RFa = RArr
        RArr = epsilon
        RExpm = epsilon
        RExpa = epsilon
        RExpConcat = epsilon
        RExpRUm = 'opr1' ExpConcat RExpRUm
        [012, 013] (0037,       opr1)
            {<}
        ExpConcat = Expa RExpConcat
        Expa = Expm RExpa
        Expm = ExpU RExpm
        ExpU = Fa
        Fa = 'id' RFa
        [012, 015] (0000,         id) {
            limite}
        RFa = RArr
        RArr = epsilon
        RExpm = epsilon
        RExpa = epsilon
        RExpConcat = epsilon
        RExpRUm = epsilon
        RExpRDois = epsilon
        RExpBoolAnd = epsilon
        RExp = epsilon
        [012, 021] (0028,          ))
            {)}
        [012, 023] (0003,         do) {
            do}

13        when (i > 2) do

        Instrucao = Comando RInstrucao
        Comando = When Comando
        When = 'when' '(' Exp ')' 'do'
            Otherwise 'end'
        [013, 007] (0008,       when) {
            when}
        [013, 012] (0027,          ()
            {(}
        Exp = ExpBoolAnd RExp
        ExpBoolAnd = ExpRDois RExpBoolAnd
        ExpRDois = ExpRUm RExpRDois
        ExpRUm = ExpConcat RExpRUm
        ExpConcat = Expa RExpConcat
        Expa = Expm RExpa
        Expm = ExpU RExpm
        ExpU = Fa
        Fa = 'id' RFa
        [013, 013] (0000,         id) {
            i}
        RFa = RArr
        RArr = epsilon
        RExpm = epsilon
        RExpa = epsilon
        RExpConcat = epsilon
        RExpRUm = 'opr1' ExpConcat RExpRUm
        [013, 015] (0037,       opr1)
            {>}
        ExpConcat = Expa RExpConcat
        Expa = Expm RExpa
        Expm = ExpU RExpm
        ExpU = Fa
        Fa = Cte
        Cte = 'cteI'
        [013, 017] (0017,       cteI)
            {2}
        RExpm = epsilon
        RExpa = epsilon
        RExpConcat = epsilon
        RExpRUm = epsilon
        RExpRDois = epsilon
        RExpBoolAnd = epsilon
        RExp = epsilon
        [013, 018] (0028,          ))
            {)}
        [013, 020] (0003,         do) {
            do}

14          arr[i] = arr[i-1] + arr[i-2];

        Otherwise = Instrucao OtherwiseR
        Instrucao = Comando RInstrucao
        Comando = 'id' RArr ComandoX
            Comando
        [014, 009] (0000,         id) {
            arr}
        RArr = '[' LRArr ']'
        [014, 012] (0029,          [)
            {[}
        LRArr = Exp
        Exp = ExpBoolAnd RExp
        ExpBoolAnd = ExpRDois RExpBoolAnd
        ExpRDois = ExpRUm RExpRDois
```

```
ExpRUm = ExpConcat RExpRUm
ExpConcat = Expa RExpConcat
Expa = Expm RExpa
Expm = ExpU RExpm
ExpU = Fa
Fa = 'id' RFa
    [014, 013] (0000,         id) {
        i}
RFa = RArr
RArr = epsilon
RExpm = epsilon
RExpa = epsilon
RExpConcat = epsilon
RExpRUm = epsilon
RExpRDois = epsilon
RExpBoolAnd = epsilon
RExp = epsilon
    [014, 014] (0030,          ])
        {]}
ComandoX = '=' Exp ';'
    [014, 016] (0036,          =)
        {=}
Exp = ExpBoolAnd RExp
ExpBoolAnd = ExpRDois RExpBoolAnd
ExpRDois = ExpRUm RExpRDois
ExpRUm = ExpConcat RExpRUm
ExpConcat = Expa RExpConcat
Expa = Expm RExpa
Expm = ExpU RExpm
ExpU = Fa
Fa = 'id' RFa
    [014, 018] (0000,         id) {
        arr}
RFa = RArr
RArr = '[' LRArr ']'
    [014, 021] (0029,          [)
        {[}
LRArr = Exp
Exp = ExpBoolAnd RExp
ExpBoolAnd = ExpRDois RExpBoolAnd
ExpRDois = ExpRUm RExpRDois
ExpRUm = ExpConcat RExpRUm
ExpConcat = Expa RExpConcat
Expa = Expm RExpa
Expm = ExpU RExpm
ExpU = Fa
Fa = 'id' RFa
    [014, 022] (0000,         id) {
        i}
RFa = RArr
RArr = epsilon
RExpm = epsilon
RExpa = 'opa' Expm RExpa
    [014, 023] (0034,        opa)
        {-}
Expm = ExpU RExpm
ExpU = Fa
Fa = Cte
Cte = 'cteI'
    [014, 024] (0017,       cteI)
        {1}
RExpm = epsilon
RExpa = epsilon
RExpConcat = epsilon
RExpRUm = epsilon
RExpRDois = epsilon
RExpBoolAnd = epsilon
RExp = epsilon
    [014, 025] (0030,          ])
        {]}
RExpm = epsilon
RExpa = 'opa' Expm RExpa
    [014, 027] (0034,        opa)
        {+}
Expm = ExpU RExpm
ExpU = Fa
Fa = 'id' RFa
    [014, 029] (0000,         id) {
        arr}
RFa = RArr
RArr = '[' LRArr ']'
    [014, 032] (0029,          [)
        {[}
LRArr = Exp
Exp = ExpBoolAnd RExp
ExpBoolAnd = ExpRDois RExpBoolAnd
ExpRDois = ExpRUm RExpRDois
ExpRUm = ExpConcat RExpRUm
ExpConcat = Expa RExpConcat
Expa = Expm RExpa
Expm = ExpU RExpm
ExpU = Fa
Fa = 'id' RFa
    [014, 033] (0000,         id) {
        i}
RFa = RArr
RArr = epsilon
RExpm = epsilon
RExpa = 'opa' Expm RExpa
    [014, 034] (0034,        opa)
        {-}
Expm = ExpU RExpm
ExpU = Fa
Fa = Cte
Cte = 'cteI'
    [014, 035] (0017,       cteI)
        {2}
RExpm = epsilon
RExpa = epsilon
RExpConcat = epsilon
RExpRUm = epsilon
RExpRDois = epsilon
RExpBoolAnd = epsilon
RExp = epsilon
    [014, 036] (0030,          ])
        {]}
RExpm = epsilon
RExpa = epsilon
RExpConcat = epsilon
RExpRUm = epsilon
RExpRDois = epsilon
RExpBoolAnd = epsilon
RExp = epsilon
    [014, 037] (0023,          ;)
        {;}
```

```
15        end

        Comando = epsilon
        RInstrucao = epsilon
        OtherwiseR = epsilon
            [015, 007] (0004,        end) {
                end}

17        print(arr[i]);

        Comando = 'id' RArr ComandoX
            Comando
            [017, 007] (0025,         id) {
                print}
        RArr = epsilon
        ComandoX = '(' ParFunc ')' ';'
            [017, 012] (0027,          () 
                {(}
        ParFunc = RParFunc
        RParFunc = Exp TRParFunc
        Exp = ExpBoolAnd RExp
        ExpBoolAnd = ExpRDois RExpBoolAnd
        ExpRDois = ExpRUm RExpRDois
        ExpRUm = ExpConcat RExpRUm
        ExpConcat = Expa RExpConcat
        Expa = Expm RExpa
        Expm = ExpU RExpm
        ExpU = Fa
        Fa = 'id' RFa
            [017, 013] (0000,         id) {
                arr}
        RFa = RArr
        RArr = '[' LRArr ']'
            [017, 016] (0029,          []
                {[}
        LRArr = Exp
        Exp = ExpBoolAnd RExp
        ExpBoolAnd = ExpRDois RExpBoolAnd
        ExpRDois = ExpRUm RExpRDois
        ExpRUm = ExpConcat RExpRUm
        ExpConcat = Expa RExpConcat
        Expa = Expm RExpa
        Expm = ExpU RExpm
        ExpU = Fa
        Fa = 'id' RFa
            [017, 017] (0000,         id) {
                i}
        RFa = RArr
        RArr = epsilon
        RExpm = epsilon
        RExpa = epsilon
        RExpConcat = epsilon
        RExpRUm = epsilon
        RExpRDois = epsilon
        RExpBoolAnd = epsilon
        RExp = epsilon
            [017, 018] (0030,          ])
                {]}
        RExpm = epsilon
        RExpa = epsilon
        RExpConcat = epsilon
        RExpRUm = epsilon
        RExpRDois = epsilon
        RExpBoolAnd = epsilon
        RExp = epsilon
        TRParFunc = epsilon
            [017, 019] (0028,          ))
                {)}
            [017, 020] (0023,          ;)
                {;}

19        when(i != limite-1) do

        Comando = When Comando
        When = 'when' '(' Exp ')' 'do'
            Otherwise 'end'
            [019, 007] (0008,       when) {
                when}
            [019, 011] (0027,          ()
                {(}
        Exp = ExpBoolAnd RExp
        ExpBoolAnd = ExpRDois RExpBoolAnd
        ExpRDois = ExpRUm RExpRDois
        ExpRUm = ExpConcat RExpRUm
        ExpConcat = Expa RExpConcat
        Expa = Expm RExpa
        Expm = ExpU RExpm
        ExpU = Fa
        Fa = 'id' RFa
            [019, 012] (0000,         id) {
                i}
        RFa = RArr
        RArr = epsilon
        RExpm = epsilon
        RExpa = epsilon
        RExpConcat = epsilon
        RExpRUm = epsilon
        RExpRDois = 'opr2' ExpRUm
            [019, 014] (0038,       opr2)
                {!=}
        ExpRUm = ExpConcat RExpRUm
        ExpConcat = Expa RExpConcat
        Expa = Expm RExpa
        Expm = ExpU RExpm
        ExpU = Fa
        Fa = 'id' RFa
            [019, 017] (0000,         id) {
                limite}
        RFa = RArr
        RArr = epsilon
        RExpm = epsilon
        RExpa = 'opa' Expm RExpa
            [019, 023] (0034,        opa)
                {-}
        Expm = ExpU RExpm
        ExpU = Fa
        Fa = Cte
        Cte = 'cteI'
            [019, 024] (0017,       cteI)
                {1}
        RExpm = epsilon
        RExpa = epsilon
        RExpConcat = epsilon
        RExpRUm = epsilon
        RExpBoolAnd = epsilon
        RExp = epsilon
```

```
        [019, 025] (0028,          ))
            {)}
        [019, 027] (0003,         do) {
            do}

20      print(", ");

    Otherwise = Instrucao OtherwiseR
    Instrucao = Comando RInstrucao
    Comando = 'id' RArr ComandoX
        Comando
        [020, 009] (0025,         id) {
            print}
    RArr = epsilon
    ComandoX = '(' ParFunc ')' ';'
        [020, 014] (0027,          ()
            {(}
    ParFunc = RParFunc
    RParFunc = Exp TRParFunc
    Exp = ExpBoolAnd RExp
    ExpBoolAnd = ExpRDois RExpBoolAnd
    ExpRDois = ExpRUm RExpRDois
    ExpRUm = ExpConcat RExpRUm
    ExpConcat = Expa RExpConcat
    Expa = Expm RExpa
    Expm = ExpU RExpm
    ExpU = Fa
    Fa = Cte
    Cte = 'cteStr'
        [020, 015] (0019,     cteStr)
            {, }
    RExpm = epsilon
    RExpa = epsilon
    RExpConcat = epsilon
    RExpRUm = epsilon
    RExpRDois = epsilon
    RExpBoolAnd = epsilon
    RExp = epsilon
    TRParFunc = epsilon
        [020, 019] (0028,          ))
            {)}
        [020, 020] (0023,          ;)
            {;}

21    end

    Comando = epsilon
    RInstrucao = epsilon
    OtherwiseR = epsilon
        [021, 007] (0004,        end) {
            end}

23    i = i + 1;

    Comando = 'id' RArr ComandoX
        Comando
        [023, 007] (0000,         id) {
            i}
    RArr = epsilon
    ComandoX = '=' Exp ';'
        [023, 009] (0036,          =)
            {=}
    Exp = ExpBoolAnd RExp
        ExpBoolAnd = ExpRDois RExpBoolAnd
        ExpRDois = ExpRUm RExpRDois
        ExpRUm = ExpConcat RExpRUm
        ExpConcat = Expa RExpConcat
        Expa = Expm RExpa
        Expm = ExpU RExpm
        ExpU = Fa
        Fa = 'id' RFa
            [023, 011] (0000,         id) {
                i}
        RFa = RArr
        RArr = epsilon
        RExpm = epsilon
        RExpa = 'opa' Expm RExpa
            [023, 013] (0034,        opa)
                {+}
        Expm = ExpU RExpm
        ExpU = Fa
        Fa = Cte
        Cte = 'cteI'
            [023, 015] (0017,       cteI)
                {1}
        RExpm = epsilon
        RExpa = epsilon
        RExpConcat = epsilon
        RExpRUm = epsilon
        RExpRDois = epsilon
        RExpBoolAnd = epsilon
        RExp = epsilon
            [023, 016] (0023,          ;)
                {;}

24    end

        Comando = epsilon
        RInstrucao = epsilon
            [024, 005] (0004,        end) {
                end}

25  end

        Comando = epsilon
        RInstrucao = epsilon
            [025, 003] (0004,        end) {
                end}

28  def void main() do

        DeclFuncoes = Funcao DeclFuncoes
        Funcao = 'def' TipoFuncao 'id' '('
            Parametro ')' 'do' Instrucao '
            end'
            [028, 003] (0002,        def) {
                def}
        TipoFuncao = Tipo FArr
        Tipo = 'void'
            [028, 007] (0011,       void) {
                void}
        FArr = epsilon
            [028, 012] (0000,         id) {
                main}
            [028, 016] (0027,          ()
                {(}
```

```
        Parametro = epsilon
            [028, 017] (0028,          ))
                {)}
            [028, 019] (0003,         do) {
                do}

 29     int limite = read();

        Instrucao = Comando RInstrucao
        Comando = TipoFixo 'id' ComandoA
            ';' Comando
        TipoFixo = Tipo FArr
        Tipo = 'int'
            [029, 005] (0012,        int) {
                int}
        FArr = epsilon
            [029, 009] (0000,         id) {
                limite}
        ComandoA = '=' Exp
            [029, 016] (0036,          =)
                {=}
        Exp = ExpBoolAnd RExp
        ExpBoolAnd = ExpRDois RExpBoolAnd
        ExpRDois = ExpRUm RExpRDois
        ExpRUm = ExpConcat RExpRUm
        ExpConcat = Expa RExpConcat
        Expa = Expm RExpa
        Expm = ExpU RExpm
        ExpU = Fa
        Fa = 'id' RFa
            [029, 018] (0024,         id) {
                read}
        RFa = '(' ParFunc ')'
            [029, 022] (0027,          ()
                {(}
        ParFunc = epsilon
            [029, 023] (0028,          ))
                {)}
        RExpm = epsilon
        RExpa = epsilon
        RExpConcat = epsilon
        RExpRUm = epsilon
        RExpRDois = epsilon
        RExpBoolAnd = epsilon
        RExp = epsilon
            [029, 024] (0023,          ;)
                {;}

 30     fib(limite);
```

```
        Comando = 'id' RArr ComandoX
            Comando
            [030, 005] (0000,         id) {
                fib}
        RArr = epsilon
        ComandoX = '(' ParFunc ')' ';'
            [030, 008] (0027,          ()
                {(}
        ParFunc = RParFunc
        RParFunc = Exp TRParFunc
        Exp = ExpBoolAnd RExp
        ExpBoolAnd = ExpRDois RExpBoolAnd
        ExpRDois = ExpRUm RExpRDois
        ExpRUm = ExpConcat RExpRUm
        ExpConcat = Expa RExpConcat
        Expa = Expm RExpa
        Expm = ExpU RExpm
        ExpU = Fa
        Fa = 'id' RFa
            [030, 009] (0000,         id) {
                limite}
        RFa = RArr
        RArr = epsilon
        RExpm = epsilon
        RExpa = epsilon
        RExpConcat = epsilon
        RExpRUm = epsilon
        RExpRDois = epsilon
        RExpBoolAnd = epsilon
        RExp = epsilon
        TRParFunc = epsilon
            [030, 015] (0028,          ))
                {)}
            [030, 016] (0023,          ;)
                {;}

 31   end

        Comando = epsilon
        RInstrucao = epsilon
            [031, 003] (0004,        end) {
                end}

 33 endmod

        DeclFuncoes = epsilon
            [033, 001] (0005,     endmod) {
                endmod}
Aceito!
End:
```

5.3 ShellSort

```
Start:
  1 defmod Shellsort do
  2
  3   def int[] shellsort(valores : int[])
      do
  4
  5     int n = len(valores); #-- retorna
      tamanho do array valores, aqui nao eh
        comando para os tokens
  6     int h = 1;
  7
  8     until(h < n) do
  9       h = h * 3 + 1;
 10     end
 11
 12     h = h / 3;
```

```
13
14      int c;
15      int j;
16
17      until(n > 2) do
18
19        rep (i : int = h, i < n, i = i +
    1) do
20          c = valores[i];
21          j = i;
22
23          until(j >= h && valores[j - h]
    > c) do
24                  valores[j] = valores[j
    - h];
25                  j = j - h;
26          end
27
28          valores[j] = c;
29        end
30
31        h = h / 2;
32
33      end
34
35      return valores;
36    end
37
38    def void main() do
39
40      int[] valores;
41
42      until(read() != EOF) do
43        valores[] = lastValueRead();
44      end
45
46      int[] arr = shellsort(valores);
47
48      rep(i : int = 0, i < len(valores),
    i = i + 1) do
49        print(arr[i]);
50        print("\n");
51      end
52
53    end
54
55  endmod
```

```
1   defmod Shellsort do

        S = 'defmod' 'id' 'do' Global
            DeclFuncoes 'endmod'
            [001, 001] (0001,     defmod) {
                defmod}
            [001, 008] (0000,         id) {
                Shellsort}
            [001, 018] (0003,         do) {
                do}

3     def int[] shellsort(valores : int[])
    do

        Global = RDeclVar
        RDeclVar = epsilon
        DeclFuncoes = Funcao DeclFuncoes
        Funcao = 'def' TipoFuncao 'id' '('
            Parametro ')' 'do' Instrucao '
            end'
            [003, 003] (0002,        def) {
                def}
        TipoFuncao = Tipo FArr
        Tipo = 'int'
            [003, 007] (0012,        int) {
                int}
        FArr = '[' RFArr ']'
            [003, 010] (0029,         []
                {[}
        RFArr = epsilon
            [003, 011] (0030,         ])
                {]}
            [003, 013] (0000,         id) {
                shellsort}
            [003, 022] (0027,         ()
                {(}
        Parametro = 'id' ':' TipoFixo
            Parametro
            [003, 023] (0000,         id) {
                valores}
            [003, 031] (0022,         :)
                {:}
        TipoFixo = Tipo FArr
        Tipo = 'int'
            [003, 033] (0012,        int) {
                int}
        FArr = '[' RFArr ']'
            [003, 036] (0029,         []
                {[}
        RFArr = epsilon
            [003, 037] (0030,         ])
                {]}
        Parametro = epsilon
            [003, 038] (0028,         ))
                {)}
            [003, 040] (0003,         do) {
                do}

5       int n = len(valores); #-- retorna
    tamanho do array valores, aqui nao eh
    comando para os tokens

        Instrucao = Comando RInstrucao
        Comando = TipoFixo 'id' ComandoA
            ';' Comando
        TipoFixo = Tipo FArr
        Tipo = 'int'
            [005, 005] (0012,        int) {
                int}
        FArr = epsilon
            [005, 009] (0000,         id) {
                n}
        ComandoA = '=' Exp
            [005, 011] (0036,         =)
                {=}
        Exp = ExpBoolAnd RExp
        ExpBoolAnd = ExpRDois RExpBoolAnd
        ExpRDois = ExpRUm RExpRDois
```

```
        ExpRUm = ExpConcat RExpRUm
        ExpConcat = Expa RExpConcat
        Expa = Expm RExpa
        Expm = ExpU RExpm
        ExpU = Fa
        Fa = 'id' RFa
            [005, 013] (0000,          id) {
               len}
        RFa = '(' ParFunc ')'
            [005, 016] (0027,           ()
               {(}
        ParFunc = RParFunc
        RParFunc = Exp TRParFunc
        Exp = ExpBoolAnd RExp
        ExpBoolAnd = ExpRDois RExpBoolAnd
        ExpRDois = ExpRUm RExpRDois
        ExpRUm = ExpConcat RExpRUm
        ExpConcat = Expa RExpConcat
        Expa = Expm RExpa
        Expm = ExpU RExpm
        ExpU = Fa
        Fa = 'id' RFa
            [005, 017] (0000,          id) {
               valores}
        RFa = RArr
        RArr = epsilon
        RExpm = epsilon
        RExpa = epsilon
        RExpConcat = epsilon
        RExpRUm = epsilon
        RExpRDois = epsilon
        RExpBoolAnd = epsilon
        RExp = epsilon
        TRParFunc = epsilon
            [005, 024] (0028,           ))
               {)}
        RExpm = epsilon
        RExpa = epsilon
        RExpConcat = epsilon
        RExpRUm = epsilon
        RExpRDois = epsilon
        RExpBoolAnd = epsilon
        RExp = epsilon
            [005, 025] (0023,           ;)
               {;}

6       int h = 1;

        Comando = TipoFixo 'id' ComandoA
            ';' Comando
        TipoFixo = Tipo FArr
        Tipo = 'int'
            [006, 005] (0012,         int) {
               int}
        FArr = epsilon
            [006, 009] (0000,          id) {
               h}
        ComandoA = '=' Exp
            [006, 011] (0036,           =)
               {=}
        Exp = ExpBoolAnd RExp
        ExpBoolAnd = ExpRDois RExpBoolAnd
        ExpRDois = ExpRUm RExpRDois
        ExpRUm = ExpConcat RExpRUm
        ExpConcat = Expa RExpConcat
        Expa = Expm RExpa
        Expm = ExpU RExpm
        ExpU = Fa
        Fa = Cte
        Cte = 'cteI'
            [006, 013] (0017,         cteI)
               {1}
        RExpm = epsilon
        RExpa = epsilon
        RExpConcat = epsilon
        RExpRUm = epsilon
        RExpRDois = epsilon
        RExpBoolAnd = epsilon
        RExp = epsilon
            [006, 014] (0023,           ;)
               {;}

8       until(h < n) do

        Comando = Until Comando
        Until = 'until' '(' Exp ')' 'do'
            Instrucao 'end'
            [008, 005] (0006,        until) {
               until}
            [008, 010] (0027,           ()
               {(}
        Exp = ExpBoolAnd RExp
        ExpBoolAnd = ExpRDois RExpBoolAnd
        ExpRDois = ExpRUm RExpRDois
        ExpRUm = ExpConcat RExpRUm
        ExpConcat = Expa RExpConcat
        Expa = Expm RExpa
        Expm = ExpU RExpm
        ExpU = Fa
        Fa = 'id' RFa
            [008, 011] (0000,          id) {
               h}
        RFa = RArr
        RArr = epsilon
        RExpm = epsilon
        RExpa = epsilon
        RExpConcat = epsilon
        RExpRUm = 'opr1' ExpConcat RExpRUm
            [008, 013] (0037,         opr1)
               {<}
        ExpConcat = Expa RExpConcat
        Expa = Expm RExpa
        Expm = ExpU RExpm
        ExpU = Fa
        Fa = 'id' RFa
            [008, 015] (0000,          id) {
               n}
        RFa = RArr
        RArr = epsilon
        RExpm = epsilon
        RExpa = epsilon
        RExpConcat = epsilon
        RExpRUm = epsilon
        RExpRDois = epsilon
        RExpBoolAnd = epsilon
        RExp = epsilon
```

```
        [008, 016] (0028,          ))
            {)}
        [008, 018] (0003,         do) {
            do}

 9        h = h * 3 + 1;

        Instrucao = Comando RInstrucao
        Comando = 'id' RArr ComandoX
            Comando
            [009, 007] (0000,         id) {
                h}
        RArr = epsilon
        ComandoX = '=' Exp ';'
            [009, 009] (0036,          =)
                {=}
        Exp = ExpBoolAnd RExp
        ExpBoolAnd = ExpRDois RExpBoolAnd
        ExpRDois = ExpRUm RExpRDois
        ExpRUm = ExpConcat RExpRUm
        ExpConcat = Expa RExpConcat
        Expa = Expm RExpa
        Expm = ExpU RExpm
        ExpU = Fa
        Fa = 'id' RFa
            [009, 011] (0000,         id) {
                h}
        RFa = RArr
        RArr = epsilon
        RExpm = 'opm' ExpU RExpm
            [009, 013] (0035,        opm)
                {*}
        ExpU = Fa
        Fa = Cte
        Cte = 'cteI'
            [009, 015] (0017,       cteI)
                {3}
        RExpm = epsilon
        RExpa = 'opa' Expm RExpa
            [009, 017] (0034,        opa)
                {+}
        Expm = ExpU RExpm
        ExpU = Fa
        Fa = Cte
        Cte = 'cteI'
            [009, 019] (0017,       cteI)
                {1}
        RExpm = epsilon
        RExpa = epsilon
        RExpConcat = epsilon
        RExpRUm = epsilon
        RExpRDois = epsilon
        RExpBoolAnd = epsilon
        RExp = epsilon
            [009, 020] (0023,          ;)
                {;}

10      end

        Comando = epsilon
        RInstrucao = epsilon
            [010, 005] (0004,        end) {
                end}
```

```
12      h = h / 3;

        Comando = 'id' RArr ComandoX
            Comando
            [012, 005] (0000,         id) {
                h}
        RArr = epsilon
        ComandoX = '=' Exp ';'
            [012, 007] (0036,          =)
                {=}
        Exp = ExpBoolAnd RExp
        ExpBoolAnd = ExpRDois RExpBoolAnd
        ExpRDois = ExpRUm RExpRDois
        ExpRUm = ExpConcat RExpRUm
        ExpConcat = Expa RExpConcat
        Expa = Expm RExpa
        Expm = ExpU RExpm
        ExpU = Fa
        Fa = 'id' RFa
            [012, 009] (0000,         id) {
                h}
        RFa = RArr
        RArr = epsilon
        RExpm = 'opm' ExpU RExpm
            [012, 011] (0035,        opm)
                {/}
        ExpU = Fa
        Fa = Cte
        Cte = 'cteI'
            [012, 013] (0017,       cteI)
                {3}
        RExpm = epsilon
        RExpa = epsilon
        RExpConcat = epsilon
        RExpRUm = epsilon
        RExpRDois = epsilon
        RExpBoolAnd = epsilon
        RExp = epsilon
            [012, 014] (0023,          ;)
                {;}

14      int c;

        Comando = TipoFixo 'id' ComandoA
            ';' Comando
        TipoFixo = Tipo FArr
        Tipo = 'int'
            [014, 005] (0012,        int) {
                int}
        FArr = epsilon
            [014, 009] (0000,         id) {
                c}
        ComandoA = epsilon
            [014, 010] (0023,          ;)
                {;}

15      int j;

        Comando = TipoFixo 'id' ComandoA
            ';' Comando
        TipoFixo = Tipo FArr
        Tipo = 'int'
```

```
        [015, 005] (0012,        int) {
            int}
    FArr = epsilon
        [015, 009] (0000,         id) {
            j}
    ComandoA = epsilon
        [015, 010] (0023,          ;)
            {;}

17  until(n > 2) do

    Comando = Until Comando
    Until = 'until' '(' Exp ')' 'do'
        Instrucao 'end'
        [017, 005] (0006,      until) {
            until}
        [017, 010] (0027,          ()
            {(}
    Exp = ExpBoolAnd RExp
    ExpBoolAnd = ExpRDois RExpBoolAnd
    ExpRDois = ExpRUm RExpRDois
    ExpRUm = ExpConcat RExpRUm
    ExpConcat = Expa RExpConcat
    Expa = Expm RExpa
    Expm = ExpU RExpm
    ExpU = Fa
    Fa = 'id' RFa
        [017, 011] (0000,         id) {
            n}
    RFa = RArr
    RArr = epsilon
    RExpm = epsilon
    RExpa = epsilon
    RExpConcat = epsilon
    RExpRUm = 'opr1' ExpConcat RExpRUm
        [017, 013] (0037,       opr1)
            {>}
    ExpConcat = Expa RExpConcat
    Expa = Expm RExpa
    Expm = ExpU RExpm
    ExpU = Fa
    Fa = Cte
    Cte = 'cteI'
        [017, 015] (0017,       cteI)
            {2}
    RExpm = epsilon
    RExpa = epsilon
    RExpConcat = epsilon
    RExpRUm = epsilon
    RExpRDois = epsilon
    RExpBoolAnd = epsilon
    RExp = epsilon
        [017, 016] (0028,          ))
            {)}
        [017, 018] (0003,         do) {
            do}

19      rep (i : int = h, i < n, i = i +
    1) do

    Instrucao = Comando RInstrucao
    Comando = Rep Comando
    Rep = 'rep' '(' VarControl ',' Exp
        ',' Id '=' Exp ')' 'do'
        Instrucao 'end'
        [019, 007] (0007,        rep) {
            rep}
        [019, 011] (0027,          ()
            {(}
    VarControl = 'id' ':' Tipo '=' Exp
        [019, 012] (0000,         id) {
            i}
        [019, 014] (0022,          :)
            {:}
    Tipo = 'int'
        [019, 016] (0012,        int) {
            int}
        [019, 020] (0036,          =)
            {=}
    Exp = ExpBoolAnd RExp
    ExpBoolAnd = ExpRDois RExpBoolAnd
    ExpRDois = ExpRUm RExpRDois
    ExpRUm = ExpConcat RExpRUm
    ExpConcat = Expa RExpConcat
    Expa = Expm RExpa
    Expm = ExpU RExpm
    ExpU = Fa
    Fa = 'id' RFa
        [019, 022] (0000,         id) {
            h}
    RFa = RArr
    RArr = epsilon
    RExpm = epsilon
    RExpa = epsilon
    RExpConcat = epsilon
    RExpRUm = epsilon
    RExpRDois = epsilon
    RExpBoolAnd = epsilon
    RExp = epsilon
        [019, 023] (0021,          ,)
            {,}
    Exp = ExpBoolAnd RExp
    ExpBoolAnd = ExpRDois RExpBoolAnd
    ExpRDois = ExpRUm RExpRDois
    ExpRUm = ExpConcat RExpRUm
    ExpConcat = Expa RExpConcat
    Expa = Expm RExpa
    Expm = ExpU RExpm
    ExpU = Fa
    Fa = 'id' RFa
        [019, 025] (0000,         id) {
            i}
    RFa = RArr
    RArr = epsilon
    RExpm = epsilon
    RExpa = epsilon
    RExpConcat = epsilon
    RExpRUm = 'opr1' ExpConcat RExpRUm
        [019, 027] (0037,       opr1)
            {<}
    ExpConcat = Expa RExpConcat
    Expa = Expm RExpa
    Expm = ExpU RExpm
    ExpU = Fa
    Fa = 'id' RFa
```

```
    [019, 029] (0000,          id) {
        n}
RFa = RArr
RArr = epsilon
RExpm = epsilon
RExpa = epsilon
RExpConcat = epsilon
RExpRUm = epsilon
RExpRDois = epsilon
RExpBoolAnd = epsilon
RExp = epsilon
    [019, 030] (0021,           ,)
        {,}
Id = 'id' Arr
    [019, 032] (0000,          id) {
        i}
Arr = epsilon
    [019, 034] (0036,           =)
        {=}
Exp = ExpBoolAnd RExp
ExpBoolAnd = ExpRDois RExpBoolAnd
ExpRDois = ExpRUm RExpRDois
ExpRUm = ExpConcat RExpRUm
ExpConcat = Expa RExpConcat
Expa = Expm RExpa
Expm = ExpU RExpm
ExpU = Fa
Fa = 'id' RFa
    [019, 036] (0000,          id) {
        i}
RFa = RArr
RArr = epsilon
RExpm = epsilon
RExpa = 'opa' Expm RExpa
    [019, 038] (0034,         opa)
        {+}
Expm = ExpU RExpm
ExpU = Fa
Fa = Cte
Cte = 'cteI'
    [019, 040] (0017,        cteI)
        {1}
RExpm = epsilon
RExpa = epsilon
RExpConcat = epsilon
RExpRUm = epsilon
RExpRDois = epsilon
RExpBoolAnd = epsilon
RExp = epsilon
    [019, 041] (0028,           ))
        {)}
    [019, 043] (0003,          do) {
        do}

20      c = valores[i];

Instrucao = Comando RInstrucao
Comando = 'id' RArr ComandoX
    Comando
    [020, 009] (0000,          id) {
        c}
RArr = epsilon
ComandoX = '=' Exp ';'
    [020, 011] (0036,           =)
        {=}
Exp = ExpBoolAnd RExp
ExpBoolAnd = ExpRDois RExpBoolAnd
ExpRDois = ExpRUm RExpRDois
ExpRUm = ExpConcat RExpRUm
ExpConcat = Expa RExpConcat
Expa = Expm RExpa
Expm = ExpU RExpm
ExpU = Fa
Fa = 'id' RFa
    [020, 013] (0000,          id) {
        valores}
RFa = RArr
RArr = '[' LRArr ']'
    [020, 020] (0029,           [)
        {[}
LRArr = Exp
Exp = ExpBoolAnd RExp
ExpBoolAnd = ExpRDois RExpBoolAnd
ExpRDois = ExpRUm RExpRDois
ExpRUm = ExpConcat RExpRUm
ExpConcat = Expa RExpConcat
Expa = Expm RExpa
Expm = ExpU RExpm
ExpU = Fa
Fa = 'id' RFa
    [020, 021] (0000,          id) {
        i}
RFa = RArr
RArr = epsilon
RExpm = epsilon
RExpa = epsilon
RExpConcat = epsilon
RExpRUm = epsilon
RExpRDois = epsilon
RExpBoolAnd = epsilon
RExp = epsilon
    [020, 022] (0030,           ])
        {]}
RExpm = epsilon
RExpa = epsilon
RExpConcat = epsilon
RExpRUm = epsilon
RExpRDois = epsilon
RExpBoolAnd = epsilon
RExp = epsilon
    [020, 023] (0023,           ;)
        {;}

21      j = i;

Comando = 'id' RArr ComandoX
    Comando
    [021, 009] (0000,          id) {
        j}
RArr = epsilon
ComandoX = '=' Exp ';'
    [021, 011] (0036,           =)
        {=}
Exp = ExpBoolAnd RExp
ExpBoolAnd = ExpRDois RExpBoolAnd
ExpRDois = ExpRUm RExpRDois
```

```
        ExpRUm = ExpConcat RExpRUm
        ExpConcat = Expa RExpConcat
        Expa = Expm RExpa
        Expm = ExpU RExpm
        ExpU = Fa
        Fa = 'id' RFa
            [021, 013] (0000,          id) {
               i}
        RFa = RArr
        RArr = epsilon
        RExpm = epsilon
        RExpa = epsilon
        RExpConcat = epsilon
        RExpRUm = epsilon
        RExpRDois = epsilon
        RExpBoolAnd = epsilon
        RExp = epsilon
            [021, 014] (0023,           ;)
               {;}

23          until(j >= h && valores[j - h]
    > c) do

        Comando = Until Comando
        Until = 'until' '(' Exp ')' 'do'
            Instrucao 'end'
            [023, 009] (0006,       until) {
               until}
            [023, 014] (0027,           ()
               {(}
        Exp = ExpBoolAnd RExp
        ExpBoolAnd = ExpRDois RExpBoolAnd
        ExpRDois = ExpRUm RExpRDois
        ExpRUm = ExpConcat RExpRUm
        ExpConcat = Expa RExpConcat
        Expa = Expm RExpa
        Expm = ExpU RExpm
        ExpU = Fa
        Fa = 'id' RFa
            [023, 015] (0000,          id) {
               j}
        RFa = RArr
        RArr = epsilon
        RExpm = epsilon
        RExpa = epsilon
        RExpConcat = epsilon
        RExpRUm = 'opr1' ExpConcat RExpRUm
            [023, 016] (0037,        opr1)
               {>=}
        ExpConcat = Expa RExpConcat
        Expa = Expm RExpa
        Expm = ExpU RExpm
        ExpU = Fa
        Fa = 'id' RFa
            [023, 019] (0000,          id) {
               h}
        RFa = RArr
        RArr = epsilon
        RExpm = epsilon
        RExpa = epsilon
        RExpConcat = epsilon
        RExpRUm = epsilon
        RExpRDois = epsilon
        RExpBoolAnd = '&&' ExpRDois
            RExpBoolAnd
            [023, 021] (0039,          &&)
               {&&}
        ExpRDois = ExpRUm RExpRDois
        ExpRUm = ExpConcat RExpRUm
        ExpConcat = Expa RExpConcat
        Expa = Expm RExpa
        Expm = ExpU RExpm
        ExpU = Fa
        Fa = 'id' RFa
            [023, 024] (0000,          id) {
               valores}
        RFa = RArr
        RArr = '[' LRArr ']'
            [023, 031] (0029,           [)
               {[}
        LRArr = Exp
        Exp = ExpBoolAnd RExp
        ExpBoolAnd = ExpRDois RExpBoolAnd
        ExpRDois = ExpRUm RExpRDois
        ExpRUm = ExpConcat RExpRUm
        ExpConcat = Expa RExpConcat
        Expa = Expm RExpa
        Expm = ExpU RExpm
        ExpU = Fa
        Fa = 'id' RFa
            [023, 032] (0000,          id) {
               j}
        RFa = RArr
        RArr = epsilon
        RExpm = epsilon
        RExpa = 'opa' Expm RExpa
            [023, 034] (0034,        opa)
               {-}
        Expm = ExpU RExpm
        ExpU = Fa
        Fa = 'id' RFa
            [023, 036] (0000,          id) {
               h}
        RFa = RArr
        RArr = epsilon
        RExpm = epsilon
        RExpa = epsilon
        RExpConcat = epsilon
        RExpRUm = epsilon
        RExpRDois = epsilon
        RExpBoolAnd = epsilon
        RExp = epsilon
            [023, 037] (0030,           ])
               {]}
        RExpm = epsilon
        RExpa = epsilon
        RExpConcat = epsilon
        RExpRUm = 'opr1' ExpConcat RExpRUm
            [023, 039] (0037,        opr1)
               {>}
        ExpConcat = Expa RExpConcat
        Expa = Expm RExpa
        Expm = ExpU RExpm
        ExpU = Fa
        Fa = 'id' RFa
```

```
            [023, 041] (0000,          id) {
                c}
        RFa = RArr
        RArr = epsilon
        RExpm = epsilon
        RExpa = epsilon
        RExpConcat = epsilon
        RExpRUm = epsilon
        RExpRDois = epsilon
        RExpBoolAnd = epsilon
        RExp = epsilon
            [023, 042] (0028,           ))
                {)}
            [023, 044] (0003,          do) {
                do}

24                  valores[j] = valores[j
    - h];

        Instrucao = Comando RInstrucao
        Comando = 'id' RArr ComandoX
            Comando
            [024, 017] (0000,          id) {
                valores}
        RArr = '[' LRArr ']'
            [024, 024] (0029,           [)
                {[}
        LRArr = Exp
        Exp = ExpBoolAnd RExp
        ExpBoolAnd = ExpRDois RExpBoolAnd
        ExpRDois = ExpRUm RExpRDois
        ExpRUm = ExpConcat RExpRUm
        ExpConcat = Expa RExpConcat
        Expa = Expm RExpa
        Expm = ExpU RExpm
        ExpU = Fa
        Fa = 'id' RFa
            [024, 025] (0000,          id) {
                j}
        RFa = RArr
        RArr = epsilon
        RExpm = epsilon
        RExpa = epsilon
        RExpConcat = epsilon
        RExpRUm = epsilon
        RExpRDois = epsilon
        RExpBoolAnd = epsilon
        RExp = epsilon
            [024, 026] (0030,           ])
                {]}
        ComandoX = '=' Exp ';'
            [024, 028] (0036,           =)
                {=}
        Exp = ExpBoolAnd RExp
        ExpBoolAnd = ExpRDois RExpBoolAnd
        ExpRDois = ExpRUm RExpRDois
        ExpRUm = ExpConcat RExpRUm
        ExpConcat = Expa RExpConcat
        Expa = Expm RExpa
        Expm = ExpU RExpm
        ExpU = Fa
        Fa = 'id' RFa
            [024, 030] (0000,          id) {
                valores}
        RFa = RArr
        RArr = '[' LRArr ']'
            [024, 037] (0029,           [)
                {[}
        LRArr = Exp
        Exp = ExpBoolAnd RExp
        ExpBoolAnd = ExpRDois RExpBoolAnd
        ExpRDois = ExpRUm RExpRDois
        ExpRUm = ExpConcat RExpRUm
        ExpConcat = Expa RExpConcat
        Expa = Expm RExpa
        Expm = ExpU RExpm
        ExpU = Fa
        Fa = 'id' RFa
            [024, 038] (0000,          id) {
                j}
        RFa = RArr
        RArr = epsilon
        RExpm = epsilon
        RExpa = 'opa' Expm RExpa
            [024, 040] (0034,         opa)
                {-}
        Expm = ExpU RExpm
        ExpU = Fa
        Fa = 'id' RFa
            [024, 042] (0000,          id) {
                h}
        RFa = RArr
        RArr = epsilon
        RExpm = epsilon
        RExpa = epsilon
        RExpConcat = epsilon
        RExpRUm = epsilon
        RExpRDois = epsilon
        RExpBoolAnd = epsilon
        RExp = epsilon
            [024, 043] (0030,           ])
                {]}
        RExpm = epsilon
        RExpa = epsilon
        RExpConcat = epsilon
        RExpRUm = epsilon
        RExpRDois = epsilon
        RExpBoolAnd = epsilon
        RExp = epsilon
            [024, 044] (0023,           ;)
                {;}

25                  j = j - h;

        Comando = 'id' RArr ComandoX
            Comando
            [025, 017] (0000,          id) {
                j}
        RArr = epsilon
        ComandoX = '=' Exp ';'
            [025, 019] (0036,           =)
                {=}
        Exp = ExpBoolAnd RExp
        ExpBoolAnd = ExpRDois RExpBoolAnd
        ExpRDois = ExpRUm RExpRDois
```

```
        ExpRUm = ExpConcat RExpRUm
        ExpConcat = Expa RExpConcat
        Expa = Expm RExpa
        Expm = ExpU RExpm
        ExpU = Fa
        Fa = 'id' RFa
            [025, 021] (0000,         id) {
                j}
        RFa = RArr
        RArr = epsilon
        RExpm = epsilon
        RExpa = 'opa' Expm RExpa
            [025, 023] (0034,        opa)
                {-}
        Expm = ExpU RExpm
        ExpU = Fa
        Fa = 'id' RFa
            [025, 025] (0000,         id) {
                h}
        RFa = RArr
        RArr = epsilon
        RExpm = epsilon
        RExpa = epsilon
        RExpConcat = epsilon
        RExpRUm = epsilon
        RExpRDois = epsilon
        RExpBoolAnd = epsilon
        RExp = epsilon
            [025, 026] (0023,          ;)
                {;}

26          end

        Comando = epsilon
        RInstrucao = epsilon
            [026, 009] (0004,        end) {
                end}

28          valores[j] = c;

        Comando = 'id' RArr ComandoX
            Comando
            [028, 009] (0000,         id) {
                valores}
        RArr = '[' LRArr ']'
            [028, 016] (0029,          [)
                {[}
        LRArr = Exp
        Exp = ExpBoolAnd RExp
        ExpBoolAnd = ExpRDois RExpBoolAnd
        ExpRDois = ExpRUm RExpRDois
        ExpRUm = ExpConcat RExpRUm
        ExpConcat = Expa RExpConcat
        Expa = Expm RExpa
        Expm = ExpU RExpm
        ExpU = Fa
        Fa = 'id' RFa
            [028, 017] (0000,         id) {
                j}
        RFa = RArr
        RArr = epsilon
        RExpm = epsilon
        RExpa = epsilon
        RExpConcat = epsilon
        RExpRUm = epsilon
        RExpRDois = epsilon
        RExpBoolAnd = epsilon
        RExp = epsilon
            [028, 018] (0030,          ])
                {]}
        ComandoX = '=' Exp ';'
            [028, 020] (0036,          =)
                {=}
        Exp = ExpBoolAnd RExp
        ExpBoolAnd = ExpRDois RExpBoolAnd
        ExpRDois = ExpRUm RExpRDois
        ExpRUm = ExpConcat RExpRUm
        ExpConcat = Expa RExpConcat
        Expa = Expm RExpa
        Expm = ExpU RExpm
        ExpU = Fa
        Fa = 'id' RFa
            [028, 022] (0000,         id) {
                c}
        RFa = RArr
        RArr = epsilon
        RExpm = epsilon
        RExpa = epsilon
        RExpConcat = epsilon
        RExpRUm = epsilon
        RExpRDois = epsilon
        RExpBoolAnd = epsilon
        RExp = epsilon
            [028, 023] (0023,          ;)
                {;}

29        end

        Comando = epsilon
        RInstrucao = epsilon
            [029, 007] (0004,        end) {
                end}

31        h = h / 2;

        Comando = 'id' RArr ComandoX
            Comando
            [031, 007] (0000,         id) {
                h}
        RArr = epsilon
        ComandoX = '=' Exp ';'
            [031, 009] (0036,          =)
                {=}
        Exp = ExpBoolAnd RExp
        ExpBoolAnd = ExpRDois RExpBoolAnd
        ExpRDois = ExpRUm RExpRDois
        ExpRUm = ExpConcat RExpRUm
        ExpConcat = Expa RExpConcat
        Expa = Expm RExpa
        Expm = ExpU RExpm
        ExpU = Fa
        Fa = 'id' RFa
            [031, 011] (0000,         id) {
                h}
        RFa = RArr
        RArr = epsilon
```

```
        RExpm = 'opm' ExpU RExpm
            [031, 013] (0035,         opm)
                {/}
        ExpU = Fa
        Fa = Cte
        Cte = 'cteI'
            [031, 015] (0017,        cteI)
                {2}
        RExpm = epsilon
        RExpa = epsilon
        RExpConcat = epsilon
        RExpRUm = epsilon
        RExpRDois = epsilon
        RExpBoolAnd = epsilon
        RExp = epsilon
            [031, 016] (0023,           ;)
                {;}

33      end

        Comando = epsilon
        RInstrucao = epsilon
            [033, 005] (0004,         end) {
                end}

35      return valores;

        Comando = Return
        Return = 'return' Exp ';'
            [035, 005] (0009,      return) {
                return}
        Exp = ExpBoolAnd RExp
        ExpBoolAnd = ExpRDois RExpBoolAnd
        ExpRDois = ExpRUm RExpRDois
        ExpRUm = ExpConcat RExpRUm
        ExpConcat = Expa RExpConcat
        Expa = Expm RExpa
        Expm = ExpU RExpm
        ExpU = Fa
        Fa = 'id' RFa
            [035, 012] (0000,          id) {
                valores}
        RFa = RArr
        RArr = epsilon
        RExpm = epsilon
        RExpa = epsilon
        RExpConcat = epsilon
        RExpRUm = epsilon
        RExpRDois = epsilon
        RExpBoolAnd = epsilon
        RExp = epsilon
            [035, 019] (0023,           ;)
                {;}

36    end

        RInstrucao = epsilon
            [036, 003] (0004,         end) {
                end}

38    def void main() do

        DeclFuncoes = Funcao DeclFuncoes

        Funcao = 'def' TipoFuncao 'id' '('
            Parametro ')' 'do' Instrucao '
            end'
            [038, 003] (0002,         def) {
                def}
        TipoFuncao = Tipo FArr
        Tipo = 'void'
            [038, 007] (0011,        void) {
                void}
        FArr = epsilon
            [038, 012] (0000,          id) {
                main}
            [038, 016] (0027,           ()
                {(}
        Parametro = epsilon
            [038, 017] (0028,           ))
                {)}
            [038, 019] (0003,          do) {
                do}

40      int[] valores;

        Instrucao = Comando RInstrucao
        Comando = TipoFixo 'id' ComandoA
            ';' Comando
        TipoFixo = Tipo FArr
        Tipo = 'int'
            [040, 005] (0012,         int) {
                int}
        FArr = '[' RFArr ']'
            [040, 008] (0029,           []
                {[}
        RFArr = epsilon
            [040, 009] (0030,           ])
                {]}
            [040, 011] (0000,          id) {
                valores}
        ComandoA = epsilon
            [040, 018] (0023,           ;)
                {;}

42      until(read() != EOF) do

        Comando = Until Comando
        Until = 'until' '(' Exp ')' 'do'
            Instrucao 'end'
            [042, 005] (0006,       until) {
                until}
            [042, 010] (0027,           ()
                {(}
        Exp = ExpBoolAnd RExp
        ExpBoolAnd = ExpRDois RExpBoolAnd
        ExpRDois = ExpRUm RExpRDois
        ExpRUm = ExpConcat RExpRUm
        ExpConcat = Expa RExpConcat
        Expa = Expm RExpa
        Expm = ExpU RExpm
        ExpU = Fa
        Fa = 'id' RFa
            [042, 011] (0024,          id) {
                read}
        RFa = '(' ParFunc ')'
```

```
    [042, 015] (0027,          ()
        {(}
ParFunc = epsilon
    [042, 016] (0028,          ))
        {)}
RExpm = epsilon
RExpa = epsilon
RExpConcat = epsilon
RExpRUm = epsilon
RExpRDois = 'opr2' ExpRUm
    [042, 018] (0038,       opr2)
        {!=}
ExpRUm = ExpConcat RExpRUm
ExpConcat = Expa RExpConcat
Expa = Expm RExpa
Expm = ExpU RExpm
ExpU = Fa
Fa = 'id' RFa
    [042, 021] (0000,         id) {
        EOF}
RFa = RArr
RArr = epsilon
RExpm = epsilon
RExpa = epsilon
RExpConcat = epsilon
RExpRUm = epsilon
RExpBoolAnd = epsilon
RExp = epsilon
    [042, 024] (0028,          ))
        {)}
    [042, 026] (0003,         do) {
        do}

43    valores[] = lastValueRead();

Instrucao = Comando RInstrucao
Comando = 'id' RArr ComandoX
    Comando
    [043, 007] (0000,         id) {
        valores}
RArr = '[' LRArr ']'
    [043, 014] (0029,          []
        {[}
LRArr = epsilon
    [043, 015] (0030,          ])
        {]}
ComandoX = '=' Exp ';'
    [043, 017] (0036,          =)
        {=}
Exp = ExpBoolAnd RExp
ExpBoolAnd = ExpRDois RExpBoolAnd
ExpRDois = ExpRUm RExpRDois
ExpRUm = ExpConcat RExpRUm
ExpConcat = Expa RExpConcat
Expa = Expm RExpa
Expm = ExpU RExpm
ExpU = Fa
Fa = 'id' RFa
    [043, 019] (0026,         id) {
        lastValueRead}
RFa = '(' ParFunc ')'
    [043, 032] (0027,          ()
        {(}
ParFunc = epsilon
    [043, 033] (0028,          ))
        {)}
RExpm = epsilon
RExpa = epsilon
RExpConcat = epsilon
RExpRUm = epsilon
RExpRDois = epsilon
RExpBoolAnd = epsilon
RExp = epsilon
    [043, 034] (0023,          ;)
        {;}

44  end

Comando = epsilon
RInstrucao = epsilon
    [044, 005] (0004,        end) {
        end}

46  int[] arr = shellsort(valores);

Comando = TipoFixo 'id' ComandoA
    ';' Comando
TipoFixo = Tipo FArr
Tipo = 'int'
    [046, 005] (0012,        int) {
        int}
FArr = '[' RFArr ']'
    [046, 008] (0029,          []
        {[}
RFArr = epsilon
    [046, 009] (0030,          ])
        {]}
    [046, 011] (0000,         id) {
        arr}
ComandoA = '=' Exp
    [046, 015] (0036,          =)
        {=}
Exp = ExpBoolAnd RExp
ExpBoolAnd = ExpRDois RExpBoolAnd
ExpRDois = ExpRUm RExpRDois
ExpRUm = ExpConcat RExpRUm
ExpConcat = Expa RExpConcat
Expa = Expm RExpa
Expm = ExpU RExpm
ExpU = Fa
Fa = 'id' RFa
    [046, 017] (0000,         id) {
        shellsort}
RFa = '(' ParFunc ')'
    [046, 026] (0027,          ()
        {(}
ParFunc = RParFunc
RParFunc = Exp TRParFunc
Exp = ExpBoolAnd RExp
ExpBoolAnd = ExpRDois RExpBoolAnd
ExpRDois = ExpRUm RExpRDois
ExpRUm = ExpConcat RExpRUm
ExpConcat = Expa RExpConcat
Expa = Expm RExpa
Expm = ExpU RExpm
ExpU = Fa
```

```
        Fa = 'id' RFa
            [046, 027] (0000,          id) {
               valores}
        RFa = RArr
        RArr = epsilon
        RExpm = epsilon
        RExpa = epsilon
        RExpConcat = epsilon
        RExpRUm = epsilon
        RExpRDois = epsilon
        RExpBoolAnd = epsilon
        RExp = epsilon
        TRParFunc = epsilon
            [046, 034] (0028,           ))
               {)}
        RExpm = epsilon
        RExpa = epsilon
        RExpConcat = epsilon
        RExpRUm = epsilon
        RExpRDois = epsilon
        RExpBoolAnd = epsilon
        RExp = epsilon
            [046, 035] (0023,           ;)
               {;}

48      rep(i : int = 0, i < len(valores),
    i = i + 1) do

        Comando = Rep Comando
        Rep = 'rep' '(' VarControl ',' Exp
            ',' Id '=' Exp ')' 'do'
            Instrucao 'end'
            [048, 005] (0007,         rep) {
               rep}
            [048, 008] (0027,           () 
               {(}
        VarControl = 'id' ':' Tipo '=' Exp
            [048, 009] (0000,          id) {
               i}
            [048, 011] (0022,           :)
               {:}
        Tipo = 'int'
            [048, 013] (0012,         int) {
               int}
            [048, 017] (0036,           =)
               {=}
        Exp = ExpBoolAnd RExp
        ExpBoolAnd = ExpRDois RExpBoolAnd
        ExpRDois = ExpRUm RExpRDois
        ExpRUm = ExpConcat RExpRUm
        ExpConcat = Expa RExpConcat
        Expa = Expm RExpa
        Expm = ExpU RExpm
        ExpU = Fa
        Fa = Cte
        Cte = 'cteI'
            [048, 019] (0017,        cteI)
               {0}
        RExpm = epsilon
        RExpa = epsilon
        RExpConcat = epsilon
        RExpRUm = epsilon
        RExpRDois = epsilon
        RExpBoolAnd = epsilon
        RExp = epsilon
            [048, 020] (0021,           ,)
               {,}
        Exp = ExpBoolAnd RExp
        ExpBoolAnd = ExpRDois RExpBoolAnd
        ExpRDois = ExpRUm RExpRDois
        ExpRUm = ExpConcat RExpRUm
        ExpConcat = Expa RExpConcat
        Expa = Expm RExpa
        Expm = ExpU RExpm
        ExpU = Fa
        Fa = 'id' RFa
            [048, 022] (0000,          id) {
               i}
        RFa = RArr
        RArr = epsilon
        RExpm = epsilon
        RExpa = epsilon
        RExpConcat = epsilon
        RExpRUm = 'opr1' ExpConcat RExpRUm
            [048, 024] (0037,        opr1)
               {<}
        ExpConcat = Expa RExpConcat
        Expa = Expm RExpa
        Expm = ExpU RExpm
        ExpU = Fa
        Fa = 'id' RFa
            [048, 026] (0000,          id) {
               len}
        RFa = '(' ParFunc ')'
            [048, 029] (0027,           () 
               {(}
        ParFunc = RParFunc
        RParFunc = Exp TRParFunc
        Exp = ExpBoolAnd RExp
        ExpBoolAnd = ExpRDois RExpBoolAnd
        ExpRDois = ExpRUm RExpRDois
        ExpRUm = ExpConcat RExpRUm
        ExpConcat = Expa RExpConcat
        Expa = Expm RExpa
        Expm = ExpU RExpm
        ExpU = Fa
        Fa = 'id' RFa
            [048, 030] (0000,          id) {
               valores}
        RFa = RArr
        RArr = epsilon
        RExpm = epsilon
        RExpa = epsilon
        RExpConcat = epsilon
        RExpRUm = epsilon
        RExpRDois = epsilon
        RExpBoolAnd = epsilon
        RExp = epsilon
        TRParFunc = epsilon
            [048, 037] (0028,           ))
               {)}
        RExpm = epsilon
        RExpa = epsilon
        RExpConcat = epsilon
        RExpRUm = epsilon
        RExpRDois = epsilon
```

```
RExpBoolAnd = epsilon
RExp = epsilon
    [048, 038] (0021,           ,)
       {,}
Id = 'id' Arr
    [048, 040] (0000,          id) {
       i}
Arr = epsilon
    [048, 042] (0036,           =)
       {=}
Exp = ExpBoolAnd RExp
ExpBoolAnd = ExpRDois RExpBoolAnd
ExpRDois = ExpRUm RExpRDois
ExpRUm = ExpConcat RExpRUm
ExpConcat = Expa RExpConcat
Expa = Expm RExpa
Expm = ExpU RExpm
ExpU = Fa
Fa = 'id' RFa
    [048, 044] (0000,          id) {
       i}
RFa = RArr
RArr = epsilon
RExpm = epsilon
RExpa = 'opa' Expm RExpa
    [048, 046] (0034,         opa)
       {+}
Expm = ExpU RExpm
ExpU = Fa
Fa = Cte
Cte = 'cteI'
    [048, 048] (0017,        cteI)
       {1}
RExpm = epsilon
RExpa = epsilon
RExpConcat = epsilon
RExpRUm = epsilon
RExpRDois = epsilon
RExpBoolAnd = epsilon
RExp = epsilon
    [048, 049] (0028,           ))
       {)}
    [048, 051] (0003,          do) {
       do}

49      print(arr[i]);

Instrucao = Comando RInstrucao
Comando = 'id' RArr ComandoX
   Comando
    [049, 007] (0025,          id) {
       print}
RArr = epsilon
ComandoX = '(' ParFunc ')' ';'
    [049, 012] (0027,           ()
       {(}
ParFunc = RParFunc
RParFunc = Exp TRParFunc
Exp = ExpBoolAnd RExp
ExpBoolAnd = ExpRDois RExpBoolAnd
ExpRDois = ExpRUm RExpRDois
ExpRUm = ExpConcat RExpRUm
ExpConcat = Expa RExpConcat
Expa = Expm RExpa
Expm = ExpU RExpm
ExpU = Fa
Fa = 'id' RFa
    [049, 013] (0000,          id) {
       arr}
RFa = RArr
RArr = '[' LRArr ']'
    [049, 016] (0029,           [)
       {[}
LRArr = Exp
Exp = ExpBoolAnd RExp
ExpBoolAnd = ExpRDois RExpBoolAnd
ExpRDois = ExpRUm RExpRDois
ExpRUm = ExpConcat RExpRUm
ExpConcat = Expa RExpConcat
Expa = Expm RExpa
Expm = ExpU RExpm
ExpU = Fa
Fa = 'id' RFa
    [049, 017] (0000,          id) {
       i}
RFa = RArr
RArr = epsilon
RExpm = epsilon
RExpa = epsilon
RExpConcat = epsilon
RExpRUm = epsilon
RExpRDois = epsilon
RExpBoolAnd = epsilon
RExp = epsilon
    [049, 018] (0030,           ])
       {]}
RExpm = epsilon
RExpa = epsilon
RExpConcat = epsilon
RExpRUm = epsilon
RExpRDois = epsilon
RExpBoolAnd = epsilon
RExp = epsilon
TRParFunc = epsilon
    [049, 019] (0028,           ))
       {)}
    [049, 020] (0023,           ;)
       {;}

50      print("\n");

Comando = 'id' RArr ComandoX
   Comando
    [050, 007] (0025,          id) {
       print}
RArr = epsilon
ComandoX = '(' ParFunc ')' ';'
    [050, 012] (0027,           ()
       {(}
ParFunc = RParFunc
RParFunc = Exp TRParFunc
Exp = ExpBoolAnd RExp
ExpBoolAnd = ExpRDois RExpBoolAnd
ExpRDois = ExpRUm RExpRDois
ExpRUm = ExpConcat RExpRUm
ExpConcat = Expa RExpConcat
```

```
        Expa = Expm RExpa
        Expm = ExpU RExpm
        ExpU = Fa
        Fa = Cte
        Cte = 'cteStr'
            [050, 013] (0019,     cteStr)
                {\n}
        RExpm = epsilon
        RExpa = epsilon
        RExpConcat = epsilon
        RExpRUm = epsilon
        RExpRDois = epsilon
        RExpBoolAnd = epsilon
        RExp = epsilon
        TRParFunc = epsilon
            [050, 017] (0028,          ))
                {)}
            [050, 018] (0023,          ;)
                {;}

51      end
```

```
          Comando = epsilon
          RInstrucao = epsilon
              [051, 005] (0004,        end) {
                  end}

  53    end

          Comando = epsilon
          RInstrucao = epsilon
              [053, 003] (0004,        end) {
                  end}

  55  endmod

          DeclFuncoes = epsilon
              [055, 001] (0005,     endmod) {
                  endmod}
Aceito!
End:
```