Universidade Federal de Alagoas
Instituto de Computação
Compiladores

Onicla - Saída dos Programas

Ramon Basto Callado Lima Lopes
José Rubens da Silva Brito

Maceió
2020.1

Sumário

1. Programas

1.1. Alô Mundo

1.1.1. Entrada

```
Function Integer Main ( ) Begin
        Print('Alo mundo!');
        Refound 0;
End
```

1.1.2. Saída

```
 1  Function Integer Main ( ) Begin
            [0001, 0009] (0022,            PR_FUNC) {Function}
            [0001, 0017] (0028,         PR_INTEGER) {Integer}
            [0001, 0022] (0042,            PR_MAIN) {Main}
            [0001, 0024] (0043,             AB_PAR) {(}
            [0001, 0026] (0044,            FEC_PAR) {)}
            [0001, 0032] (0039,           PR_BEGIN) {Begin}
  2      Print('Alo mundo!');
            [0002, 0007] (0034,           PR_PRINT) {Print}
            [0002, 0008] (0043,             AB_PAR) {(}
            [0002, 0020] (0005,         CTE_CADCHA) {'Alo mundo!'}
            [0002, 0021] (0044,            FEC_PAR) {)}
            [0002, 0022] (0047,           TERMINAL) {;}
  3      Refound 0;
            [0003, 0009] (0023,         PR_REFOUND) {Refound}
            [0003, 0011] (0003,            CTE_INT) {0}
            [0003, 0012] (0047,           TERMINAL) {;}
  4  End
            [0004, 0004] (0040,             PR_END) {End}
            [0004, 0005] (0000,               EOF) {EOF}
```

1.2. Série de Fibonacci

1.2.1. Entrada

```
Function Integer fibonacci(Integer n) Begin
                If(n < 2) Begin
                        Refound n;
                End
                Else Begin
                        Refound fibonacci(n - 1) + fibonacci(n - 2);
```

```
        End
    End

    Function Integer Main ( ) Begin
        Integer n, i;
        Print('Digite o tamanho da sequencia: ');
        Input(n);

        While(i <= n) Begin
            Print(fibonacci(i) ^ ' ');
            i = i + 1;
        End

        Refound 0;
    End
```

Saída

```
1  Function Integer fibonacci(Integer n) Begin
        [0001, 0009] (0022,            PR_FUNC) {Function}
        [0001, 0017] (0028,         PR_INTEGER) {Integer}
        [0001, 0027] (0001,                ID) {fibonacci}
        [0001, 0028] (0043,            AB_PAR) {(}
        [0001, 0035] (0028,         PR_INTEGER) {Integer}
        [0001, 0037] (0001,                ID) {n}
        [0001, 0038] (0044,            FEC_PAR) {)}
        [0001, 0044] (0039,          PR_BEGIN) {Begin}
 2          If(n < 2) Begin
        [0002, 0005] (0024,             PR_IF) {If}
        [0002, 0006] (0043,            AB_PAR) {(}
        [0002, 0007] (0001,                ID) {n}
        [0002, 0009] (0015,           OP_LESS) {<}
        [0002, 0011] (0003,           CTE_INT) {2}
        [0002, 0012] (0044,            FEC_PAR) {)}
        [0002, 0018] (0039,          PR_BEGIN) {Begin}
 3              Refound n;
        [0003, 0011] (0023,         PR_REFOUND) {Refound}
        [0003, 0013] (0001,                ID) {n}
        [0003, 0014] (0047,          TERMINAL) {;}
 4          End
        [0004, 0006] (0040,             PR_END) {End}
 5          Else Begin
        [0005, 0007] (0025,            PR_ELSE) {Else}
        [0005, 0013] (0039,          PR_BEGIN) {Begin}
```

```
6                   Refound fibonacci(n - 1) + fibonacci(n - 2);
      [0006, 0011] (0023,          PR_REFOUND) {Refound}
      [0006, 0021] (0001,                ID) {fibonacci}
      [0006, 0022] (0043,             AB_PAR) {(}
      [0006, 0023] (0001,                ID) {n}
      [0006, 0025] (0009,             OP_SUB) {-}
      [0006, 0027] (0003,            CTE_INT) {1}
      [0006, 0028] (0044,            FEC_PAR) {)}
      [0006, 0030] (0008,             OP_AD) {+}
      [0006, 0040] (0001,                ID) {fibonacci}
      [0006, 0041] (0043,             AB_PAR) {(}
      [0006, 0042] (0001,                ID) {n}
      [0006, 0044] (0009,             OP_SUB) {-}
      [0006, 0046] (0003,            CTE_INT) {2}
      [0006, 0047] (0044,            FEC_PAR) {)}
      [0006, 0048] (0047,           TERMINAL) {;}
7           End
      [0007, 0006] (0040,             PR_END) {End}
8     End
      [0008, 0005] (0040,             PR_END) {End}
9
10    Function Integer Main ( ) Begin
      [0010, 0010] (0022,             PR_FUNC) {Function}
      [0010, 0018] (0028,          PR_INTEGER) {Integer}
      [0010, 0023] (0042,             PR_MAIN) {Main}
      [0010, 0025] (0043,             AB_PAR) {(}
      [0010, 0027] (0044,            FEC_PAR) {)}
      [0010, 0033] (0039,           PR_BEGIN) {Begin}
11          Integer n, i;
      [0011, 0010] (0028,          PR_INTEGER) {Integer}
      [0011, 0012] (0001,                ID) {n}
      [0011, 0013] (0048,               SEP) {,}
      [0011, 0015] (0001,                ID) {i}
      [0011, 0016] (0047,           TERMINAL) {;}
12          Print('Digite o tamanho da sequencia: ');
      [0012, 0008] (0034,           PR_PRINT) {Print}
      [0012, 0009] (0043,             AB_PAR) {(}
      [0012, 0016] (0052,          ER_KEYWORD) {Digite}
      [0012, 0018] (0001,                ID) {o}
      [0012, 0026] (0001,                ID) {tamanho}
      [0012, 0029] (0001,                ID) {da}
      [0012, 0039] (0001,                ID) {sequencia}
      [0012, 0043] (0044,            FEC_PAR) {)}
      [0012, 0044] (0047,           TERMINAL) {;}
```

```
13          Input(n);
      [0013, 0008] (0033,          PR_INPUT) {Input}
      [0013, 0009] (0043,           AB_PAR) {(}
      [0013, 0010] (0001,              ID) {n}
      [0013, 0011] (0044,           FEC_PAR) {)}
      [0013, 0012] (0047,          TERMINAL) {;}
14
15          While(i <= n) Begin
      [0015, 0008] (0026,          PR_WHILE) {While}
      [0015, 0009] (0043,           AB_PAR) {(}
      [0015, 0010] (0001,              ID) {i}
      [0015, 0013] (0017,         OP_LESSEQ) {<=}
      [0015, 0015] (0001,              ID) {n}
      [0015, 0016] (0044,           FEC_PAR) {)}
      [0015, 0022] (0039,          PR_BEGIN) {Begin}
16                Print(fibonacci(i) ^ ‘ ’);
      [0016, 0009] (0034,          PR_PRINT) {Print}
      [0016, 0010] (0043,           AB_PAR) {(}
      [0016, 0019] (0001,              ID) {fibonacci}
      [0016, 0020] (0043,           AB_PAR) {(}
      [0016, 0021] (0001,              ID) {i}
      [0016, 0022] (0044,           FEC_PAR) {)}
      [0016, 0024] (0013,         OP_CONCAT) {^}
      [0016, 0029] (0044,           FEC_PAR) {)}
      [0016, 0030] (0047,          TERMINAL) {;}
17                i = i + 1;
      [0017, 0005] (0001,              ID) {i}
      [0017, 0007] (0006,           OP_ATR) {=}
      [0017, 0009] (0001,              ID) {i}
      [0017, 0011] (0008,           OP_AD) {+}
      [0017, 0013] (0003,          CTE_INT) {1}
      [0017, 0014] (0047,          TERMINAL) {;}
18          End
      [0018, 0006] (0040,           PR_END) {End}
19
20          Refound 0;
      [0020, 0010] (0023,        PR_REFOUND) {Refound}
      [0020, 0012] (0003,          CTE_INT) {0}
      [0020, 0013] (0047,          TERMINAL) {;}
21    End
      [0021, 0005] (0040,           PR_END) {End}
      [0021, 0006] (0000,             EOF) {EOF}
```

1.3. Shellsort

1.3.1. Entrada

```
Function Void shellsort(Integer array[ ], Integer n) Begin
	Integer h = 1, c, j;

	While (h < n) Begin
		h = h * 3 + 1;
	End

	h = h / 3;

	While(h > 0) Begin
		Repeat (Integer i = h, 1, n) Begin
			c = array[i];
			j = i;
			While (j >= h And array[j - h] > c) Begin
				array[j] = array[j - h];
				j = j - h;
			End
			array[j] = c;
		End
		h = h / 2;
	End
	Refound;
End

Function Integer Main ( ) Begin
	Integer n;
	Print('Digite o tamanho do array a ser ordenado: ');
	Input(n);
	Integer array[n];

	Print('Digite aleatoriamente os numero para serem ordenados: ');
	Repeat (Integer i = 0, 1, n) Begin
		Input(array[i]);
	End
	Print('Valores adicionados: ');
	Repeat (Integer i = 0, 1, n) Begin
		Print(array[i]);
	End
	shellsort(array[n], n);
```

```
        Print('Valores ordenados: ');
        Repeat (Integer i = 0, 1, n) Begin
                Print(array[i] ^ ' ');
        End

        Refound 0;
End
```

1.3.2. Saída

```
1  Function Void shellsort(Integer array[ ], Integer n) Begin
            [0001, 0009] (0022,            PR_FUNC) {Function}
            [0001, 0014] (0041,            PR_VOID) {Void}
            [0001, 0024] (0001,                ID) {shellsort}
            [0001, 0025] (0043,            AB_PAR) {(}
            [0001, 0032] (0028,         PR_INTEGER) {Integer}
            [0001, 0038] (0001,                ID) {array}
            [0001, 0039] (0045,            AB_COL) {[}
            [0001, 0041] (0046,            FEC_COL) {]}
            [0001, 0042] (0048,               SEP) {,}
            [0001, 0050] (0028,         PR_INTEGER) {Integer}
            [0001, 0052] (0001,                ID) {n}
            [0001, 0053] (0044,            FEC_PAR) {)}
            [0001, 0059] (0039,           PR_BEGIN) {Begin}
  2               Integer h = 1, c, j;
            [0002, 0010] (0028,         PR_INTEGER) {Integer}
            [0002, 0012] (0001,                ID) {h}
            [0002, 0014] (0006,            OP_ATR) {=}
            [0002, 0016] (0003,           CTE_INT) {1}
            [0002, 0017] (0048,               SEP) {,}
            [0002, 0019] (0001,                ID) {c}
            [0002, 0020] (0048,               SEP) {,}
            [0002, 0022] (0001,                ID) {j}
            [0002, 0023] (0047,          TERMINAL) {;}
  3
  4  While (h < n) Begin
            [0004, 0006] (0026,           PR_WHILE) {While}
            [0004, 0008] (0043,            AB_PAR) {(}
            [0004, 0009] (0001,                ID) {h}
            [0004, 0011] (0015,           OP_LESS) {<}
            [0004, 0013] (0001,                ID) {n}
            [0004, 0014] (0044,            FEC_PAR) {)}
            [0004, 0020] (0039,           PR_BEGIN) {Begin}
  5                     h = h * 3 + 1;
```

```
        [0005, 0005] (0001,              ID) {h}
        [0005, 0007] (0006,           OP_ATR) {=}
        [0005, 0009] (0001,              ID) {h}
        [0005, 0011] (0010,         OP_MULT) {*}
        [0005, 0013] (0003,         CTE_INT) {3}
        [0005, 0015] (0008,          OP_AD) {+}
        [0005, 0017] (0003,         CTE_INT) {1}
        [0005, 0018] (0047,        TERMINAL) {;}
6           End
        [0006, 0006] (0040,          PR_END) {End}
7
8           h = h / 3;
        [0008, 0004] (0001,              ID) {h}
        [0008, 0006] (0006,           OP_ATR) {=}
        [0008, 0008] (0001,              ID) {h}
        [0008, 0010] (0011,          OP_DIV) {/}
        [0008, 0012] (0003,         CTE_INT) {3}
        [0008, 0013] (0047,        TERMINAL) {;}
9
10          While(h > 0) Begin
        [0010, 0008] (0026,        PR_WHILE) {While}
        [0010, 0009] (0043,          AB_PAR) {(}
        [0010, 0010] (0001,              ID) {h}
        [0010, 0012] (0014,      OP_GREATER) {>}
        [0010, 0014] (0003,         CTE_INT) {0}
        [0010, 0015] (0044,          FEC_PAR) {)}
        [0010, 0021] (0039,        PR_BEGIN) {Begin}
11              Repeat (Integer i = h, 1, n) Begin
        [0011, 0010] (0027,       PR_REPEAT) {Repeat}
        [0011, 0012] (0043,          AB_PAR) {(}
        [0011, 0019] (0028,      PR_INTEGER) {Integer}
        [0011, 0021] (0001,              ID) {i}
        [0011, 0023] (0006,           OP_ATR) {=}
        [0011, 0025] (0001,              ID) {h}
        [0011, 0026] (0048,             SEP) {,}
        [0011, 0028] (0003,         CTE_INT) {1}
        [0011, 0029] (0048,             SEP) {,}
        [0011, 0031] (0001,              ID) {n}
        [0011, 0032] (0044,          FEC_PAR) {)}
        [0011, 0038] (0039,        PR_BEGIN) {Begin}
12  c = array[i];
        [0012, 0002] (0001,              ID) {c}
        [0012, 0004] (0006,           OP_ATR) {=}
        [0012, 0010] (0001,              ID) {array}
```

```
        [0012, 0011] (0045,            AB_COL) {[}
        [0012, 0012] (0001,               ID) {i}
        [0012, 0013] (0046,           FEC_COL) {]}
        [0012, 0014] (0047,          TERMINAL) {;}
13                                  j = i;
        [0013, 0021] (0001,               ID) {j}
        [0013, 0023] (0006,           OP_ATR) {=}
        [0013, 0025] (0001,               ID) {i}
        [0013, 0026] (0047,          TERMINAL) {;}
14                                  While (j >= h And array[j - h] > c) Begin
        [0014, 0025] (0026,          PR_WHILE) {While}
        [0014, 0027] (0043,           AB_PAR) {(}
        [0014, 0028] (0001,               ID) {j}
        [0014, 0031] (0016,        OP_GREATEQ) {>=}
        [0014, 0033] (0001,               ID) {h}
        [0014, 0037] (0020,           PR_AND) {And}
        [0014, 0043] (0001,               ID) {array}
        [0014, 0044] (0045,           AB_COL) {[}
        [0014, 0045] (0001,               ID) {j}
        [0014, 0047] (0009,           OP_SUB) {-}
        [0014, 0049] (0001,               ID) {h}
        [0014, 0050] (0046,           FEC_COL) {]}
        [0014, 0052] (0014,         OP_GREATER) {>}
        [0014, 0054] (0001,               ID) {c}
        [0014, 0055] (0044,           FEC_PAR) {)}
        [0014, 0061] (0039,          PR_BEGIN) {Begin}
15                                      array[j] = array[j - h];
        [0015, 0030] (0001,               ID) {array}
        [0015, 0031] (0045,           AB_COL) {[}
        [0015, 0032] (0001,               ID) {j}
        [0015, 0033] (0046,           FEC_COL) {]}
        [0015, 0035] (0006,           OP_ATR) {=}
        [0015, 0041] (0001,               ID) {array}
        [0015, 0042] (0045,           AB_COL) {[}
        [0015, 0043] (0001,               ID) {j}
        [0015, 0045] (0009,           OP_SUB) {-}
        [0015, 0047] (0001,               ID) {h}
        [0015, 0048] (0046,           FEC_COL) {]}
        [0015, 0049] (0047,          TERMINAL) {;}
16                                      j = j - h;
        [0016, 0026] (0001,               ID) {j}
        [0016, 0028] (0006,           OP_ATR) {=}
        [0016, 0030] (0001,               ID) {j}
        [0016, 0032] (0009,           OP_SUB) {-}
```

```
        [0016, 0034] (0001,              ID) {h}
        [0016, 0035] (0047,         TERMINAL) {;}
17                                 End
        [0017, 0023] (0040,          PR_END) {End}
18                                 array[j] = c;
        [0018, 0025] (0001,              ID) {array}
        [0018, 0026] (0045,          AB_COL) {[}
        [0018, 0027] (0001,              ID) {j}
        [0018, 0028] (0046,         FEC_COL) {]}
        [0018, 0030] (0006,          OP_ATR) {=}
        [0018, 0032] (0001,              ID) {c}
        [0018, 0033] (0047,         TERMINAL) {;}
19                   End
        [0019, 0007] (0040,          PR_END) {End}
20                   h = h / 2;
        [0020, 0005] (0001,              ID) {h}
        [0020, 0007] (0006,          OP_ATR) {=}
        [0020, 0009] (0001,              ID) {h}
        [0020, 0011] (0011,          OP_DIV) {/}
        [0020, 0013] (0003,         CTE_INT) {2}
        [0020, 0014] (0047,         TERMINAL) {;}
21           End
        [0021, 0006] (0040,          PR_END) {End}
22           Refound;
        [0022, 0010] (0023,       PR_REFOUND) {Refound}
        [0022, 0011] (0047,         TERMINAL) {;}
23   End
        [0023, 0005] (0040,          PR_END) {End}
24
25 Function Integer Main ( ) Begin
        [0025, 0009] (0022,          PR_FUNC) {Function}
        [0025, 0017] (0028,       PR_INTEGER) {Integer}
        [0025, 0022] (0042,          PR_MAIN) {Main}
        [0025, 0024] (0043,          AB_PAR) {(}
        [0025, 0026] (0044,         FEC_PAR) {)}
        [0025, 0032] (0039,         PR_BEGIN) {Begin}
26           Integer n;
        [0026, 0010] (0028,       PR_INTEGER) {Integer}
        [0026, 0012] (0001,              ID) {n}
        [0026, 0013] (0047,         TERMINAL) {;}
27           Print('Digite o tamanho do array a ser ordenado: ');
        [0027, 0008] (0034,         PR_PRINT) {Print}
        [0027, 0009] (0043,          AB_PAR) {(}
```

```
        [0027, 0053] (0005,          CTE_CADCHA) {'Digite o tamanho do array a ser
ordenado: '}
        [0027, 0054] (0044,           FEC_PAR) {)}
        [0027, 0055] (0047,          TERMINAL) {;}
 28           Input(n);
        [0028, 0008] (0033,          PR_INPUT) {Input}
        [0028, 0009] (0043,           AB_PAR) {(}
        [0028, 0010] (0001,              ID) {n}
        [0028, 0011] (0044,           FEC_PAR) {)}
        [0028, 0012] (0047,          TERMINAL) {;}
 29           Integer array[n];
        [0029, 0010] (0028,          PR_INTEGER) {Integer}
        [0029, 0016] (0001,              ID) {array}
        [0029, 0017] (0045,           AB_COL) {[}
        [0029, 0018] (0001,              ID) {n}
        [0029, 0019] (0046,           FEC_COL) {]}
        [0029, 0020] (0047,          TERMINAL) {;}
 30
 31           Print('Digite aleatoriamente os numero para serem ordenados: ');
        [0031, 0008] (0034,          PR_PRINT) {Print}
        [0031, 0009] (0043,           AB_PAR) {(}
        [0031, 0065] (0005,          CTE_CADCHA) {'Digite aleatoriamente os
numero para serem ordenados: '}
        [0031, 0066] (0044,           FEC_PAR) {)}
        [0031, 0067] (0047,          TERMINAL) {;}
 32  Repeat (Integer i = 0, 1, n) Begin
        [0032, 0007] (0027,          PR_REPEAT) {Repeat}
        [0032, 0009] (0043,           AB_PAR) {(}
        [0032, 0016] (0028,          PR_INTEGER) {Integer}
        [0032, 0018] (0001,              ID) {i}
        [0032, 0020] (0006,           OP_ATR) {=}
        [0032, 0022] (0003,           CTE_INT) {0}
        [0032, 0023] (0048,              SEP) {,}
        [0032, 0025] (0003,           CTE_INT) {1}
        [0032, 0026] (0048,              SEP) {,}
        [0032, 0028] (0001,              ID) {n}
        [0032, 0029] (0044,           FEC_PAR) {)}
        [0032, 0035] (0039,          PR_BEGIN) {Begin}
 33     Input(array[i]);
        [0033, 0007] (0033,          PR_INPUT) {Input}
        [0033, 0008] (0043,           AB_PAR) {(}
        [0033, 0013] (0001,              ID) {array}
        [0033, 0014] (0045,           AB_COL) {[}
        [0033, 0015] (0001,              ID) {i}
```

```
        [0033, 0016] (0046,            FEC_COL) {[]}
        [0033, 0017] (0044,            FEC_PAR) {)}
        [0033, 0018] (0047,           TERMINAL) {;}
34  End
        [0034, 0004] (0040,            PR_END) {End}
35  Print('Valores adicionados: ');
        [0035, 0006] (0034,          PR_PRINT) {Print}
        [0035, 0007] (0043,            AB_PAR) {(}
        [0035, 0030] (0005,        CTE_CADCHA) {'Valores adicionados: '}
        [0035, 0031] (0044,            FEC_PAR) {)}
        [0035, 0032] (0047,           TERMINAL) {;}
36  Repeat (Integer i = 0, 1, n) Begin
        [0036, 0007] (0027,          PR_REPEAT) {Repeat}
        [0036, 0009] (0043,            AB_PAR) {(}
        [0036, 0016] (0028,        PR_INTEGER) {Integer}
        [0036, 0018] (0001,               ID) {i}
        [0036, 0020] (0006,            OP_ATR) {=}
        [0036, 0022] (0003,           CTE_INT) {0}
        [0036, 0023] (0048,               SEP) {,}
        [0036, 0025] (0003,           CTE_INT) {1}
        [0036, 0026] (0048,               SEP) {,}
        [0036, 0028] (0001,               ID) {n}
        [0036, 0029] (0044,            FEC_PAR) {)}
        [0036, 0035] (0039,          PR_BEGIN) {Begin}
37    Print(array[i]);
        [0037, 0007] (0034,          PR_PRINT) {Print}
        [0037, 0008] (0043,            AB_PAR) {(}
        [0037, 0013] (0001,               ID) {array}
        [0037, 0014] (0045,            AB_COL) {[}
        [0037, 0015] (0001,               ID) {i}
        [0037, 0016] (0046,            FEC_COL) {]}
        [0037, 0017] (0044,            FEC_PAR) {)}
        [0037, 0018] (0047,           TERMINAL) {;}
38  End
        [0038, 0004] (0040,            PR_END) {End}
39          shellsort(array[n], n);
        [0039, 0012] (0001,               ID) {shellsort}
        [0039, 0013] (0043,            AB_PAR) {(}
        [0039, 0018] (0001,               ID) {array}
        [0039, 0019] (0045,            AB_COL) {[}
        [0039, 0020] (0001,               ID) {n}
        [0039, 0021] (0046,            FEC_COL) {]}
        [0039, 0022] (0048,               SEP) {,}
        [0039, 0024] (0001,               ID) {n}
```

```
        [0039, 0025] (0044,            FEC_PAR) {)}
        [0039, 0026] (0047,           TERMINAL) {;}
40
41              Print('Valores ordenados: ');
        [0041, 0008] (0034,           PR_PRINT) {Print}
        [0041, 0009] (0043,            AB_PAR) {(}
        [0041, 0030] (0005,         CTE_CADCHA) {'Valores ordenados: '}
        [0041, 0031] (0044,            FEC_PAR) {)}
        [0041, 0032] (0047,           TERMINAL) {;}
42  Repeat (Integer i = 0, 1, n) Begin
        [0042, 0007] (0027,          PR_REPEAT) {Repeat}
        [0042, 0009] (0043,            AB_PAR) {(}
        [0042, 0016] (0028,         PR_INTEGER) {Integer}
        [0042, 0018] (0001,                ID) {i}
        [0042, 0020] (0006,            OP_ATR) {=}
        [0042, 0022] (0003,           CTE_INT) {0}
        [0042, 0023] (0048,               SEP) {,}
        [0042, 0025] (0003,           CTE_INT) {1}
        [0042, 0026] (0048,               SEP) {,}
        [0042, 0028] (0001,                ID) {n}
        [0042, 0029] (0044,            FEC_PAR) {)}
        [0042, 0035] (0039,           PR_BEGIN) {Begin}
43    Print(array[i] ^ ' ');
        [0043, 0007] (0034,           PR_PRINT) {Print}
        [0043, 0008] (0043,            AB_PAR) {(}
        [0043, 0013] (0001,                ID) {array}
        [0043, 0014] (0045,            AB_COL) {[}
        [0043, 0015] (0001,                ID) {i}
        [0043, 0016] (0046,            FEC_COL) {]}
        [0043, 0018] (0013,         OP_CONCAT) {^}
        [0043, 0022] (0004,          CTE_CHAR) {' '}
        [0043, 0023] (0044,            FEC_PAR) {)}
        [0043, 0024] (0047,           TERMINAL) {;}
44  End
        [0044, 0004] (0040,             PR_END) {End}
45
46              Refound 0;
        [0046, 0010] (0023,         PR_REFOUND) {Refound}
        [0046, 0012] (0003,           CTE_INT) {0}
        [0046, 0013] (0047,           TERMINAL) {;}


47    End
        [0047, 0005] (0040,             PR_END) {End}
        [0047, 0006] (0000,               EOF) {EOF}
```