

# Erros

-> Expressões faltantes/inválidas em comandos if/while/loop

```
etapa7 > E syerrs2.txt
1  a = char: 'a';
2  c = bool: TRUE;
3  mat = int [10];
4  v1 = int[3]: 77 'a' 0 1.4;
5
6
7  main (e=int, w=bool) = int {
8      i = i+1
9      if "ue" then {
10         | print "oh my expression"
11     }
12
13     while () {
14         | i=i+1
15     }
16
17     loop (+)
18     | print "nao sabe a sintaxe nee"
19
20
21     print "some string", "another string"
22     return 1
23 };
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE 1: bash

```
[colombelli@colombelli etapa7]$ ./etapa7 syerrs2.txt
Syntax error at line 13: wrong if expression
Syntax error at line 15: wrong expression for command while
Syntax error at line 21: wrong expression for command loop
[colombelli@colombelli etapa7]$
```

-> erros diversos (tipo indefinido, ; ou } faltante, declarações e assinalamentos errados...)

```
etapa7 > E syerrs1.txt
1  a = yey: 'a';
2  c bool: TRUE;
3  mat = int [];
4  mat = aloha [10];
5  v1 int[3]: 77 'a' 0 1.4;
6
7  foo2 (aa=) = int {};
8
9  foo3 (aa=int) = int {
10     | print "this one is right!\n"
11 };
12
13 foo4 (aa=int) = int { ;
14
15 main (e=int, w=bool) = int {
16     i = i + a
17     mat[main(a, c) + 1 + 's'] = a
18     i = a+i+5.10*0.12
19     a 1
20     print "dsgj"
21     return 10
22 }
23
24 foo1 (aa=int) = wrongType {
25     | print "my type is wrong"
26     return 0
27 };
```

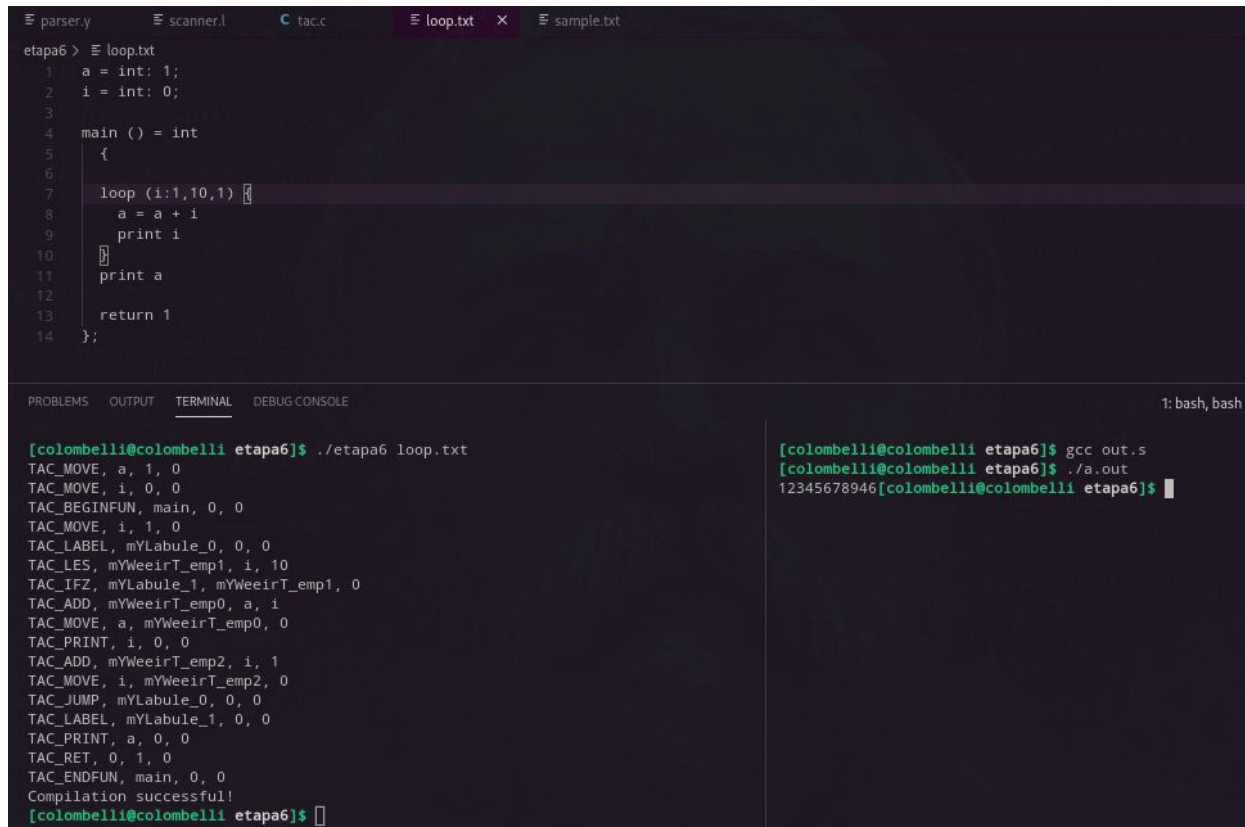
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE 1: bash

```
[colombelli@colombelli etapa7]$ ./etapa7 syerrs1.txt
Syntax error at line 1: unknown type for variable declaration
Syntax error at line 2: wrong variable declaration
Syntax error at line 3: wrong size in vector declaration
Syntax error at line 4: unknown type for vector declaration
Syntax error at line 5: wrong vector declaration
Syntax error at line 7: wrong function declaration
Syntax error at line 13: missing ;
Syntax error at line 19: expecting ;
[colombelli@colombelli etapa7]$
```

# Otimização

A técnica de otimização escolhida foi o Loop Unrolling. A otimização funciona automaticamente sempre que as expressões definidoras do loop são literais inteiros.

Exemplos antes da implementação:



The screenshot shows a code editor with a file named `loop.txt` open. The code in the editor is as follows:

```
1 a = int: 1;
2 i = int: 0;
3
4 main () = int
5 {
6
7   loop (i:1,10,1)
8     a = a + i
9     print i
10
11   print a
12
13   return 1
14 };
```

Below the editor, the terminal window shows the compilation and execution process. The left pane of the terminal shows the output of the compilation command `./etapa6 loop.txt`, which includes TAC (Three-Address Code) instructions and a successful compilation message:

```
[colombelli@colombelli etapa6]$ ./etapa6 loop.txt
TAC_MOVE, a, 1, 0
TAC_MOVE, i, 0, 0
TAC_BEGINFUN, main, 0, 0
TAC_MOVE, i, 1, 0
TAC_LABEL, mYLabule_0, 0, 0
TAC_LES, mYWeeirT_emp1, i, 10
TAC_IFZ, mYLabule_1, mYWeeirT_emp1, 0
TAC_ADD, mYWeeirT_emp0, a, i
TAC_MOVE, a, mYWeeirT_emp0, 0
TAC_PRINT, i, 0, 0
TAC_ADD, mYWeeirT_emp2, i, 1
TAC_MOVE, i, mYWeeirT_emp2, 0
TAC_JUMP, mYLabule_0, 0, 0
TAC_LABEL, mYLabule_1, 0, 0
TAC_PRINT, a, 0, 0
TAC_RET, 0, 1, 0
TAC_ENDFUN, main, 0, 0
Compilation successful!
[colombelli@colombelli etapa6]$
```

The right pane of the terminal shows the execution of the program:

```
[colombelli@colombelli etapa6]$ gcc out.s
[colombelli@colombelli etapa6]$ ./a.out
12345678946[colombelli@colombelli etapa6]$
```

À direita o resultado: impressão de i indo de 1 a 9, e o print final do a: 45

Outro exemplo de loop não otimizado, dessa vez com passo:

```
parser.y scanner.l tac.c loop.txt sample.txt
etapa6 > loop.txt
1  a = int: 1;
2  i = int: 0;
3
4  main () = int
5  {
6
7  loop [i:2,10,2] {
8      a = a + i
9      print i
10 }
11 print a
12
13 return 1
14 };

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
[colombelli@colombelli etapa6]$ ./etapa6 loop.txt
TAC_MOVE, a, 1, 0
TAC_MOVE, i, 0, 0
TAC_BEGINFUN, main, 0, 0
TAC_MOVE, i, 2, 0
TAC_LABEL, mYLabule_0, 0, 0
TAC_LES, mYWeeirT_emp1, i, 10
TAC_IFZ, mYLabule_1, mYWeeirT_emp1, 0
TAC_ADD, mYWeeirT_emp0, a, i
TAC_MOVE, a, mYWeeirT_emp0, 0
TAC_PRINT, i, 0, 0
TAC_ADD, mYWeeirT_emp2, i, 2
TAC_MOVE, i, mYWeeirT_emp2, 0
TAC_JUMP, mYLabule_0, 0, 0
TAC_LABEL, mYLabule_1, 0, 0
TAC_PRINT, a, 0, 0
TAC_RET, 0, 1, 0
TAC_ENDFUN, main, 0, 0
Compilation successful!
[colombelli@colombelli etapa6]$

[colombelli@colombelli etapa6]$ gcc out.s
[colombelli@colombelli etapa6]$ ./a.out
246821[colombelli@colombelli etapa6]$
```

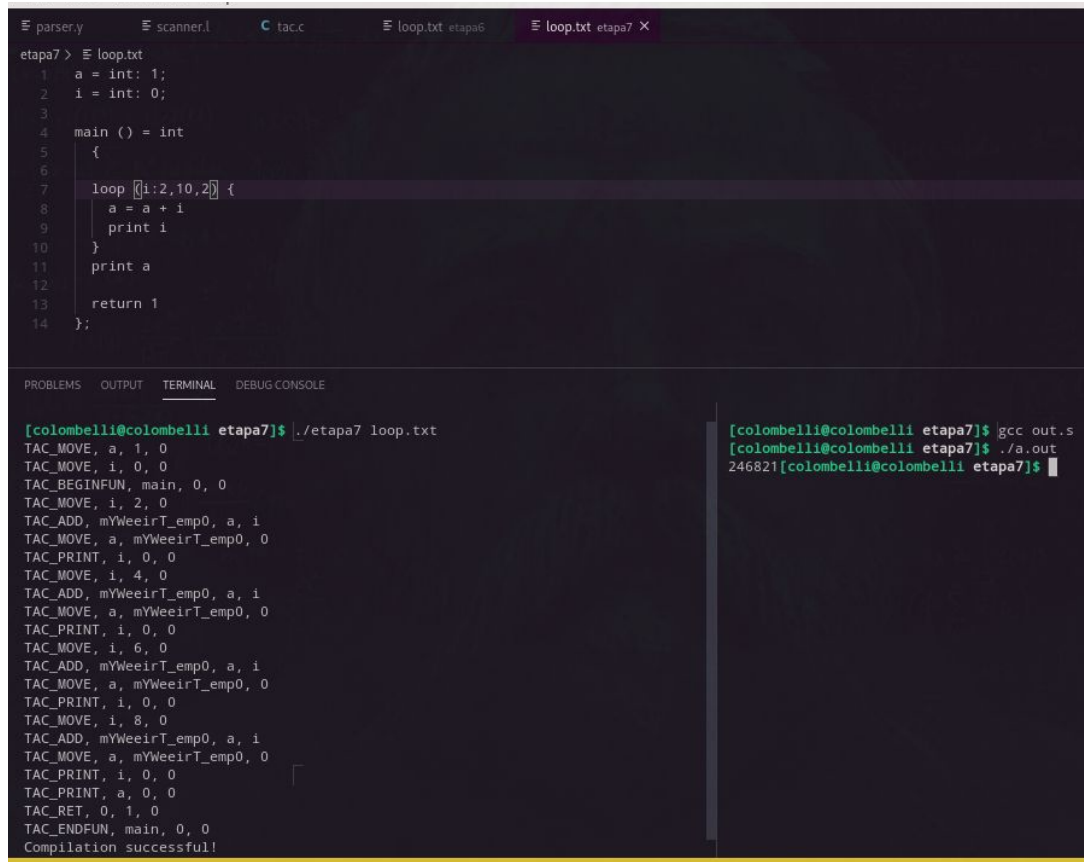
Otimização destes mesmos loops acima:

```
parser.y scanner.l tacc loop.txt etapa6 loop.txt etapa7 X
etapa7 > loop.txt
1  a = int: 1;
2  i = int: 0;
3
4  main () = int
5  {
6
7      loop (i:1,10,1) {
8          a = a + i
9          print i
10         }
11         print a
12
13         return 1
14     };

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE 1: bas

[colombelli@colombelli etapa7]$ ./etapa7 loop.txt
TAC_MOVE, a, 1, 0
TAC_MOVE, i, 0, 0
TAC_BEGINFUN, main, 0, 0
TAC_MOVE, i, 1, 0
TAC_ADD, mYWeeirT_emp0, a, i
TAC_MOVE, a, mYWeeirT_emp0, 0
TAC_PRINT, i, 0, 0
TAC_MOVE, i, 2, 0
TAC_ADD, mYWeeirT_emp0, a, i
TAC_MOVE, a, mYWeeirT_emp0, 0
TAC_PRINT, i, 0, 0
TAC_MOVE, i, 3, 0
TAC_ADD, mYWeeirT_emp0, a, i
TAC_MOVE, a, mYWeeirT_emp0, 0
TAC_PRINT, i, 0, 0
TAC_MOVE, i, 4, 0
TAC_ADD, mYWeeirT_emp0, a, i
TAC_MOVE, a, mYWeeirT_emp0, 0
TAC_PRINT, i, 0, 0
TAC_MOVE, i, 5, 0
TAC_ADD, mYWeeirT_emp0, a, i
TAC_MOVE, a, mYWeeirT_emp0, 0
TAC_PRINT, i, 0, 0

[colombelli@colombelli etapa7]$ gcc out.s
[colombelli@colombelli etapa7]$ ./a.out
12345678946[colombelli@colombelli etapa7]$
```



The screenshot shows a code editor with a dark theme. The top bar has tabs for 'parser.y', 'scanner.l', 'tac.c', 'loop.txt etapa6', and 'loop.txt etapa7 X'. The main editor area shows a C program in 'loop.txt' with line numbers 1 to 14. The code defines a variable 'a' as an integer with value 1, a variable 'i' as an integer with value 0, and a 'main' function that calls 'int main()' and enters a loop 'loop [i:2,10,2]' where 'a' is incremented and 'i' is printed. The bottom panel has tabs for 'PROBLEMS', 'OUTPUT', 'TERMINAL', and 'DEBUG CONSOLE'. The 'TERMINAL' tab is active, showing the command './etapa7 loop.txt' and its output, which lists a series of TAC instructions like 'TAC\_MOVE, a, 1, 0' and 'TAC\_ADD, mYWeeirT\_emp0, a, i'. The 'OUTPUT' tab is also active, showing the command 'gcc out.s' and its output, which displays the memory address '246821'.

```
etapa7 > loop.txt
1  a = int: 1;
2  i = int: 0;
3
4  main () = int
5  {
6
7  loop [i:2,10,2] {
8      a = a + 1
9      print i
10 }
11 print a
12
13 return 1
14 };
```

```
[colombelli@colombelli etapa7]$ ./etapa7 loop.txt
TAC_MOVE, a, 1, 0
TAC_MOVE, i, 0, 0
TAC_BEGINFUN, main, 0, 0
TAC_MOVE, i, 2, 0
TAC_ADD, mYWeeirT_emp0, a, i
TAC_MOVE, a, mYWeeirT_emp0, 0
TAC_PRINT, i, 0, 0
TAC_MOVE, i, 4, 0
TAC_ADD, mYWeeirT_emp0, a, i
TAC_MOVE, a, mYWeeirT_emp0, 0
TAC_PRINT, i, 0, 0
TAC_MOVE, i, 6, 0
TAC_ADD, mYWeeirT_emp0, a, i
TAC_MOVE, a, mYWeeirT_emp0, 0
TAC_PRINT, i, 0, 0
TAC_MOVE, i, 8, 0
TAC_ADD, mYWeeirT_emp0, a, i
TAC_MOVE, a, mYWeeirT_emp0, 0
TAC_PRINT, i, 0, 0
TAC_PRINT, a, 0, 0
TAC_RET, 0, 1, 0
TAC_ENDFUN, main, 0, 0
Compilation successful!
```

```
[colombelli@colombelli etapa7]$ gcc out.s
[colombelli@colombelli etapa7]$ ./a.out
246821[colombelli@colombelli etapa7]$
```

Outra otimização interessante implementada foi que, se for detectado um loop em que o valor inicial é maior que o final, ele só é ignorado, já que não executará o seu bloco de comandos nenhuma vez.

**Antes da otimização:**

```
parser.y scanner.l tac.c loop.txt etapa6 x loop.txt etapa7
etapa6 > loop.txt
1 a = int: 1;
2 i = int: 0;
3
4 main () = int
5 {
6
7   loop (i:10,2,2) {
8     a = a + i
9     print i
10  }
11  print a
12
13  return 1
14 };

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

[colombelli@colombelli etapa6]$ ./etapa6 loop.txt
TAC_MOVE, a, 1, 0
TAC_MOVE, i, 0, 0
TAC_BEGINFUN, main, 0, 0
TAC_MOVE, i, 10, 0
TAC_LABEL, mYLabule_0, 0, 0
TAC_LES, mYWeeirT_emp1, i, 2
TAC_IFZ, mYLabule_1, mYWeeirT_emp1, 0
TAC_ADD, mYWeeirT_emp0, a, i
TAC_MOVE, a, mYWeeirT_emp0, 0
TAC_PRINT, i, 0, 0
TAC_ADD, mYWeeirT_emp2, i, 2
TAC_MOVE, i, mYWeeirT_emp2, 0
TAC_JUMP, mYLabule_0, 0, 0
TAC_LABEL, mYLabule_1, 0, 0
TAC_PRINT, a, 0, 0
TAC_RET, 0, 1, 0
TAC_ENDFUN, main, 0, 0
Compilation successful!
[colombelli@colombelli etapa6]$

[colombelli@colombelli etapa6]$ gcc out.s
[colombelli@colombelli etapa6]$ ./a.out
1
[colombelli@colombelli etapa6]$
```

## Depois da otimização:

```
parser.y scanner.l tac.c loop.txt etapa6 loop.txt etapa7 x
etapa7 > loop.txt
1 a = int: 1;
2 i = int: 0;
3
4 main () = int
5 {
6
7   loop (i:10,2,2) {
8     a = a + i
9     print i
10  }
11  print a
12
13  return 1
14 };

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

[colombelli@colombelli etapa7]$ ./etapa7 loop.txt
TAC_MOVE, a, 1, 0
TAC_MOVE, i, 0, 0
TAC_BEGINFUN, main, 0, 0
TAC_PRINT, a, 0, 0
TAC_RET, 0, 1, 0
TAC_ENDFUN, main, 0, 0
Compilation successful!
[colombelli@colombelli etapa7]$

[colombelli@colombelli etapa7]$ gcc out.s
[colombelli@colombelli etapa7]$ ./a.out
1
[colombelli@colombelli etapa7]$
```