COMP LabBook 2024 2 - E3

Lucas M. Schnorr

November 17, 2024

Contents

1 Histórico de comentários

1.1 DONE GrupoA

1.1.1 E3

- Nenhum comentário.

1.1.2 E2

☒ Arquivo parser.y submetido na versão definitiva

☐ Poderiam documentar o comando `%locations` e o que ele faz

1.1.3 E1

☐ Para os tokens especiais, pode-se colocá-los todos na mesma regra

  – E usar yytext[0] ao invés de explicitamente usar o literal

☐ Quebra de linha é considerado espaço em branco (pode-se colocar na mesma regra)

1.2 DONE GrupoB

1.2.1 E3

☐ Veja os pontos não marcados como feitos abaixo, na E1; entendo que eles ainda estão em aberto.

1.2.2 E2

☒ É necessário atualizar o arquivo `README.md` na medida que o projeto avança

- Por exemplo, a listagem da estrutura do projeto

1.2.3 E1

- Usando Makefile como um "wrapper" do CMake (só para deixar anotado)
- Boa documentação do projeto com o arquivo `README.md`

☒ Espaços ignorados podem ser aglutinados em uma única regra

- Quebras de linha são considerados espaços

☐ Na hora de montar o pacote tgz, por favor, remover todos os arquivos "ocultos" que começam por ".". Eles não aparecem na saída do comando `ls`, mas o `tar` os vê e os inclui. Informar ao `tar` para não inclui-los.

1.3 DONE GrupoC

1.3.1 E3

☐ A estrutura do valor léxico não é um ponteiro no `%union`. OK, mas atenção às próximas etapas pois o valor léxico deverá ser "copiado" (ao invés de ter seu ponteiro gerenciado).

1.3.2 E2

☒ Funções "estranhas" (não usadas) no final do parser.y

- Exemplo, `parse_string`. Imagino que sejam para um uso alternativo de funcionamento do parser, visto que envolvem uma chamada à `yyparse`.

☒ No arquivo main.c, o include do `parser.tab.h` possui um caminho relativo. Em geral isso não é uma boa prática, sendo preferível informar ao compilador onde procurar arquivos de cabeçalhos via o parâmetro `-I` (neste caso, poderia ser algo como `-I./obj/` em algum lugar do `Makefile`).

1.3.3 E1

☒ O arquivo scanner.l, contendo código, normalmente fica em diretórios `src`

☒ O arquivo Makefile não segue a ideia de compilação separada por arquivo, algo em geral benéfico até para projetos pequenos.

- O alvo `test` não funciona, provavelmente porque o conteúdo de `src/testing` foi removido do pacote
- Pode-se criar uma arquivo `Makefile.alt` somente para testes
  - Removê-lo do tgz na hora de submeter

1.4 DONE GrupoD

1.4.1 E3

- Praticamente todos os testes falham.

- Ao invés de gerar uma única árvore, várias subárvores aparecem na saída

☐ A estrutura do valor léxico não é um ponteiro no `%union`. OK, mas atenção às próximas etapas pois o valor léxico deverá ser "copiado" (ao invés de ter seu ponteiro gerenciado).

☐ Veja os pontos não marcados como feitos abaixo, na E1; entendo que eles ainda estão em aberto.

1.4.2 E2

☒ A regra `expressao` não segue a especificação da E2 pois não implementa precedência de operadores (Sec 3.4 da E2). Para que a precedência possa ser implementada, precisas criar "níveis", por exemplo, no primeiro nível é o or, depois o and, e assim por diante. Seria algo assim:

```
expressao: expressao TK_OC_OR exp1 | exp1
exp1: exp1 TK_OC_AND exp2 | exp2
exp2: ...
```

Podemos trocar uma ideia caso ainda não tiverem entendido.

1.4.3 E1

☐ Para os tokens especiais, pode-se colocá-los todos na mesma regra

- E usar yytext[0] ao invés de explicitamente usar o literal

☐ Melhorar o makefile para que se possa usufruir de um sistema de compilação que permita compilação parcial dos vários fontes do projeto.

1.5 DONE GrupoE

1.5.1 E3

☐ A estrutura do valor léxico não é um ponteiro no `%union`. OK, mas atenção às próximas etapas pois o valor léxico deverá ser "copiado" (ao invés de ter seu ponteiro gerenciado).

1.5.2 E2

☒ O que é a "metodologia de tdd"? (comentário no `scanner.l`)

- Não respondido.

☒ A regra `expressao` não segue a especificação da E2 pois não implementa precedência de operadores (Sec 3.4 da E2). Para que a precedência possa ser implementada, precisas criar "níveis", por exemplo, no primeiro nível é o or, depois o and, e assim por diante. Seria algo assim:

```
expressao: expressao TK_OC_OR exp1 | exp1
exp1: exp1 TK_OC_AND exp2 | exp2
exp2: ...
```

Podemos trocar uma ideia caso ainda não tiverem entendido.

1.5.3 E1

☐ Para os tokens especiais, pode-se colocá-los todos na mesma regra

- E usar yytext[0] ao invés de explicitamente usar o literal

1.6 DONE GrupoF

1.6.1 E3

- Muitos testes falham.
  - Labels errados (problemas de copy/paste?)
    - Por exemplo no igual-igual
  - Mas também temos erros mais graves de uso da `asd`.

1.6.2 E2

- Vários testes falham. Olhando rapidamente, parece que o problema tem a ver com a `expressao` que deve virar um `literal` ou apenas com `literal`.

1.6.3 E1

☐ O Makefile não segue a filosofia geral para construção de projetos, pois possui listagens de código nas dependências e não possui uma regra de compilação intermediária de código objeto que permite a compilação parcial do projeto.

☒ Os espaços podem ser aglutinados em uma única regra

☒ Para os tokens especiais, pode-se colocá-los todos na mesma regra

- E usar yytext[0] ao invés de explicitamente usar o literal

1.7 DONE GrupoG

1.7.1 E3

☐ A estrutura do valor léxico não é um ponteiro no `%union`. OK, mas atenção às próximas etapas pois o valor léxico deverá ser "copiado" (ao invés de ter seu ponteiro gerenciado).

1.7.2 E2

- Nenhum comentário.

1.7.3 E1

☒ Arquivos devem estar na raiz

☒ Normalmente evita-se de #include com um caminho relativo (ou absoluto)

  – No caso farias somente #include "tokens.h", instruindo o compilador (com -I) a procurar os cabeçalhos em determinado lugar

☒ Para os tokens especiais, pode-se colocá-los todos na mesma regra

  – E usar yytext[0] ao invés de explicitamente usar o literal

☒ A regra do barra-enne é redundante com a classe [:space:]

☒ Boa organização em subdiretórios, mas o Makefile precisa melhorar visto que não é uma boa prática em receitas misturar entradas .c e .o como no alvo $(ETAPA). Além disso, a compilação de .o pode ser via regra única por intermédio de wildcards, conforme visto no tutorial indicado.

☒ Arquivos `deliver.sh`, `tester.py`, `tests` podem ser omitidos do tgz

1.8 DONE GrupoH

1.8.1 E3

☐ typedef é seu amigo para o tipo da árvore

1.8.2 E2

- Nenhum comentário.

1.8.3 E1

☐ O caractere barra-ene está incluso na classe `[:blank:]`, portanto temos uma regra redundante

☒ O Makefile não segue a filosofia geral para construção de projetos, pois possui listagens de código nas dependências e não possui uma regra de compilação intermediária de código objeto que permite a compilação parcial do projeto.

1.9 TODO GrupoI (não submetido)

1.9.1 E3

- Não submetido.

1.9.2 E2

- Arquivos devem estar na raiz

  – O peso deste comentário aumentou

1.9.3 E1

☐ Arquivos devem estar na raiz

☒ Deve-se evitar colocar a implementação de funções no cabeçalho do scanner, priorizando a última seção do arquivo ou em arquivos suplementos.

☒ Ao invés de implementar yywrap, pode-se usar a opção para desabilitar essa funcionalidade.

- [/] Alvos e receitas do makefile são majoritariamente manuais, sem wildcards. O makefile pode ficar bem mais sucinto se empregar os conhecimentos do tutorial indicado.

  – Além disto, possui regras específicas da etapa1, mesmo sabendo que temos outras etapas por vir.

1.10 DONE GrupoJ

1.10.1 E3

☐ Na hora de montar o pacote tgz, por favor, remover todos os arquivos "ocultos" que começam por ".". Eles não aparecem na saída do comando `ls`, mas o `tar` os vê e os inclui. Informar ao `tar` para não inclui-los.

☐ Veja os pontos não marcados como feitos abaixo, tanto na E2 quanto na E1; entendo que eles ainda estão em aberto.

1.10.2 E2

- Preliminar: calculou a coluna do erro sintático!

☒ No arquivo main.c, o include do `parser.tab.h` possui um caminho relativo. Em geral isso não é uma boa prática, sendo preferível informar ao compilador onde procurar arquivos de cabeçalhos via o parâmetro `-I` (neste caso, poderia ser algo como `-I./obj/` em algum lugar do `Makefile`).

☐ O arquivo `Makefile` evolui bastante, mas de uma maneira geral ele ficou muito complexo (veja o tamanho). Poderia ficar bem mais simples, sobretudo pela característica ainda pequena de nosso projeto.

☐ Veja os pontos não marcados como feitos abaixo; entendo que eles ainda estão em aberto.

1.10.3 E1

☐ As ações associadas às regras estão com identação diferente, algumas alinhadas outras não, no arquivo scanner.l

☐ O makefile parece ter código boilerplate que não se aplica neste projeto, pois ele vê se existe um subdiretório src, adaptando-se em função. Se o grupo não tem a intenção de organizar em subdiretórios, sugiro simplificar o Makefile.

☐ Não se usa a filosofia de compilação parcial dos arquivos (-c), ou seja, sempre se compila tudo novamente.

☒ A variável tar file pode ser definida a partir do nome de binary.

1.11 DONE GrupoK

1.11.1 E3

☐ A estrutura do valor léxico não é um ponteiro no `%union`. OK, mas atenção às próximas etapas pois o valor léxico deverá ser "copiado" (ao invés de ter seu ponteiro gerenciado).

1.11.2 E2

☒ Preliminar: arquivos devem estar na raiz

1.11.3 E1

☐ Não há necessidade de stdio.h no arquivo scanner.l

☒ Melhorar o makefile para que se possa usufruir de um sistema de compilação que permita compilação parcial dos vários fontes do projeto.

1.12 DONE GrupoL

1.12.1 E3

☐ Veja os pontos não marcados como feitos abaixo, na E1; entendo que eles ainda estão em aberto.

1.12.2 E2

- Nenhum comentário.

1.12.3 E1

☒ Normalmente o ; nos comandos em C ficam imediatamente após a último token do comando, sem espaços.

☒ Não há necessidade de incluir stdio.h. Além disso, cabeçalhos de bibliotecas de sistema são incluídas com <stdio.h> ao invés de "stdio.h".

☐ Melhorar o makefile para que se possa usufruir de um sistema de compilação que permita compilação parcial dos vários fontes do projeto.

☒ Evitar de empacotar o diretório `testes` (e os arquivos `quero.*\.sh`).

1.13 DONE GrupoM

1.13.1 E3

☐ A estrutura do valor léxico não é um ponteiro no `%union`. OK, mas atenção às próximas etapas pois o valor léxico deverá ser "copiado" (ao invés de ter seu ponteiro gerenciado).

☐ Nos comentários que categorizam as regras gramaticais, usar texto não em uppercase total.

☐ Veja os pontos não marcados como feitos abaixo, na E2; entendo que eles ainda estão em aberto.

1.13.2 E2

☒ Preliminar: usou o comando %left, em desacordo com a especificação E2

☐ Usar wildcards do makefile para simplificar as regras

1.13.3 E1

☒ Melhorar o makefile para que se possa usufruir de um sistema de compilação que permita compilação parcial dos vários fontes do projeto.

☒ Não há necessidade de incluir stdio.h.

☒ Os espaços ignorados podem ser aglutinados em uma única regra

1.14 DONE GrupoN

1.14.1 E3

☐ Falha de segmentação em praticamente todos os testes.

☐ O valor léxico (`yylval.valor_lexico`), conforme E3, deve ser especificado apenas para literais e identificadores. Na solução submetido, temos valor léxico para vários outros elementos desnecessários (palavras-reservadas, operadores simples e compostos, etc).

☐ Obtive um "warning: type clash on default action" na linha 132 do parser.y. Isso ocorre porque `cmdblck` tem tipo, e nenhuma ação foi incluída ali para definir o valor desse NT. A ação poderia ser algo como $$ = $2.

☐ Veja os pontos não marcados como feitos abaixo, na E2; entendo que eles ainda estão em aberto.

1.14.2 E2

☐ Melhorar o makefile para que se possa usufruir de um sistema de compilação que permita compilação parcial dos vários fontes do projeto.

1.14.3 E1

☒ Na hora de montar o pacote tgz, por favor, remover todos os arquivos "ocultos" que começam por ".". Eles não aparecem na saída do comando `ls`, mas o `tar` os vê e os inclui. Informar ao `tar` para não inclui-los.

☒ Não temos aspas simples

☒ Não temos comentários multilinha portanto não há necessidade de %x

1.15 DONE GrupoO

1.15.1 E3

☐ Veja os pontos não marcados como feitos abaixo, na E1; entendo que eles ainda estão em aberto.

1.15.2 E2

☒ A regra `expressao` não segue a especificação da E2 pois não implementa precedência de operadores (Sec 3.4 da E2). Para que a precedência possa ser implementada, precisas criar "níveis", por exemplo, no primeiro nível é o or, depois o and, e assim por diante. Seria algo assim:

```
expressao: expressao TK_OC_OR exp1 | exp1
exp1: exp1 TK_OC_AND exp2 | exp2
exp2: ...
```

Podemos trocar uma ideia caso ainda não tiverem entendido.

☒ Melhorar a identação das receitas do makefile, um tab é o suficiente.

☒ Melhorar o makefile para que se possa usufruir de um sistema de compilação que permita compilação parcial dos vários fontes do projeto.

1.15.3 E1

☒ Os espaços ignorados podem ser aglutinados em uma única regra

  - Inclusive o barra-enne poderia ser aglutinado visto que trata-se de um espaço também

☐ Comentários são legais, mas melhor se estiverem não todos em maiúscula

  - Para não confundir com constantes do código

1.16 DONE GrupoP

1.16.1 E3

☐ Temos três campos em `%union`, sendo um `asd_tree_p`. Para que ele serve? De acordo com a especificação E3, seriam somente necessários os dois primeiros campos.

☐ A estrutura do valor léxico não é um ponteiro no `%union`. OK, mas atenção às próximas etapas pois o valor léxico deverá ser "copiado" (ao invés de ter seu ponteiro gerenciado).

☐ Veja os pontos não marcados como feitos abaixo, na E1; entendo que eles ainda estão em aberto.

1.16.2 E2

☒ Melhor identificar que o número que aparece no relatório de erro é um número de linha

☒ Documentar melhor as regras da gramáticas, as agrupando e colocando espaços entre categorias de regras, de maneira a facilitar a leitura da gramática.

1.16.3 E1

☐ Melhorar o makefile para que se possa usufruir de um sistema de compilação que permita compilação parcial dos vários fontes do projeto.

1.17 TODO GrupoQ (não submetido)

1.17.1 E3

- Não submetido

1.17.2 E2

- Não submetido

1.17.3 E1

- Não submetido

1.18 DONE GrupoR

1.18.1 E3

☐ A estrutura do valor léxico não é um ponteiro no `%union`. OK, mas atenção às próximas etapas pois o valor léxico deverá ser "copiado" (ao invés de ter seu ponteiro gerenciado).

1.18.2 E2

☒ Preliminar: melhor identificar que o número que aparece no relatório de erro é um número de linha

1.18.3 E1

☒ Na função `get_line_number`, corrigir a identação.

☒ Melhorar o makefile para que se possa usufruir de um sistema de compilação que permita compilação parcial dos vários fontes do projeto.

  - Ter um alvo que empregue o parâmetro `-c`

1.19 DONE GrupoS

1.19.1 E3

☐ A estrutura do valor léxico não é um ponteiro no `%union`. OK, mas atenção às próximas etapas pois o valor léxico deverá ser "copiado" (ao invés de ter seu ponteiro gerenciado).

1.19.2 E2

☒ Nos comentários que categorizam as regras gramaticais, usar texto não em uppercase total.

1.19.3 E1

☒ Ignorar também o caractere tab com barra-t

☒ Aglutinar caracteres ignorados (espaços) em uma única regra

☒ Melhorar o makefile para que se possa usufruir de um sistema de compilação que permita compilação parcial dos vários fontes do projeto.

1.20 DONE GrupoT

1.20.1 E3

☐ Veja os pontos não marcados como feitos abaixo, na E2; entendo que eles ainda estão em aberto.

1.20.2 E2

☒ Remover o arquivo tokens.h conforme recomendado na especificação E2

☐ No makefile, a compilação de .o pode ser via regra única por intermédio de wildcards, conforme visto no tutorial indicado.

1.20.3 E1

☒ O arquivo scanner.l, contendo código, normalmente fica em diretórios `src`

☒ O arquivo Makefile não segue a ideia de compilação separada por arquivo, algo em geral benéfico até para projetos pequenos.

☒ Para os tokens especiais, pode-se colocá-los todos na mesma regra

- E usar yytext[0] ao invés de explicitamente usar o literal

1.21 DONE GrupoZ

1.21.1 E3

☐ A estrutura do valor léxico não é um ponteiro no `%union`. OK, mas atenção às próximas etapas pois o valor léxico deverá ser "copiado" (ao invés de ter seu ponteiro gerenciado).

1.21.2 E2

☒ Retomo o comentário deixado pelo grupo "TODO: Check for precedencia"

- Realmente ficou faltando.

1.21.3 E1

☒ Melhorar o makefile para que se possa usufruir de um sistema de compilação que permita compilação parcial dos vários fontes do projeto.

2 Objetivo

2.1 Sumário

Grupo	Objetivo
GrupoA	10
GrupoB	9.32
GrupoC	9.66
GrupoD	0.68
GrupoE	10
GrupoF	1.86
GrupoG	10
GrupoH	10
GrupoJ	10
GrupoK	5.93
GrupoL	9.66
GrupoM	9.83
GrupoN	0.17
GrupoO	10
GrupoP	10
GrupoR	9.83
GrupoS	10
GrupoT	9.83
GrupoZ	7.8

2.2 Análise dos diversos erros encontrados

2.2.1 Erros sintáticos ou léxicos

Os seguintes grupos tem erros sintáticos ainda que todas as entradas fornecidas estão lexicalmente e sintaticamente válidas de acordo com as especificações E1 e E2.

Grupo	test	Output

2.2.2 Erros de falha de segmentação

Os programas dos seguintes grupos terminam por falha de segmentação com as entradas especificadas.

Grupo	test	Output
GrupoN	z01	Segmentation fault
GrupoN	z02	Segmentation fault
GrupoN	z03	Segmentation fault
GrupoN	z04	Segmentation fault
GrupoN	z05	Segmentation fault
GrupoN	z06	Segmentation fault
GrupoN	z07	Segmentation fault
GrupoN	z08	Segmentation fault
GrupoN	z09	Segmentation fault
GrupoN	z10	Segmentation fault
GrupoN	z11	Segmentation fault
GrupoN	z12	Segmentation fault
GrupoN	z13	Segmentation fault
GrupoN	z14	Segmentation fault
GrupoN	z15	Segmentation fault
GrupoN	z16	Segmentation fault
GrupoN	z17	Segmentation fault
GrupoN	z18	Segmentation fault
GrupoN	z19	Segmentation fault
GrupoN	z20	Segmentation fault
GrupoN	z21	Segmentation fault
GrupoN	z22	Segmentation fault
GrupoN	z23	Segmentation fault
GrupoN	z24	Segmentation fault
GrupoN	z25	Segmentation fault
GrupoN	z26	Segmentation fault
GrupoN	z27	Segmentation fault
GrupoN	z28	Segmentation fault
GrupoN	z29	Segmentation fault
GrupoN	z30	Segmentation fault

Continued on next page

Continued from previous page

Grupo	test	Output
GrupoN	z31	Segmentation fault
GrupoN	z32	Segmentation fault
GrupoN	z33	Segmentation fault
GrupoN	z34	Segmentation fault
GrupoN	z35	Segmentation fault
GrupoN	z36	Segmentation fault
GrupoN	z37	Segmentation fault
GrupoN	z38	Segmentation fault
GrupoN	z39	Segmentation fault
GrupoN	z40	Segmentation fault
GrupoN	z41	Segmentation fault
GrupoN	z42	Segmentation fault
GrupoN	z43	Segmentation fault
GrupoN	z44	Segmentation fault
GrupoN	z45	Segmentation fault
GrupoN	z46	Segmentation fault
GrupoN	z48	Segmentation fault
GrupoN	z49	Segmentation fault
GrupoN	z50	Segmentation fault
GrupoN	z51	Segmentation fault
GrupoN	z52	Segmentation fault
GrupoN	z53	Segmentation fault
GrupoN	z54	Segmentation fault
GrupoN	z55	Segmentation fault
GrupoN	z56	Segmentation fault
GrupoN	z57	Segmentation fault
GrupoN	z58	Segmentation fault
GrupoN	z59	Segmentation fault

2.2.3 Erros de geração de árvore errada

Estes são os grupos para os quais a árvore gerada na saída não está de acordo com a árvore de referência definida na especificação da E3. O professor disponibiliza arquivos `dot` das árvores de referência (arquivos `.ref.dot`). O professor recomenda o uso da ferramenta `xdot` para visualizar as árvores (`apt install xdot`).

```
xdot - interactive viewer for Graphviz dot files
```

O professor também fornece as árvores geradas pelos grupos para cada uma das entradas (arquivos `.dot` em diretórios específicos para cada grupo). Assim, basta abrir a árvore de referência e a árvore gerada pelo grupo para entender o equívoco. Se dúvidas permanecerem, entre em contato com o professor.

Grupo	test
GrupoB	z03
GrupoB	z06
GrupoB	z08
GrupoB	z51
GrupoC	z00
GrupoC	z07
GrupoD	z01
GrupoD	z02
GrupoD	z03
GrupoD	z04
GrupoD	z05
GrupoD	z06
GrupoD	z07
GrupoD	z08
GrupoD	z09
GrupoD	z10
GrupoD	z11
GrupoD	z12
GrupoD	z13
GrupoD	z14
GrupoD	z15
GrupoD	z16
GrupoD	z17
GrupoD	z18

Continued on next page

Continued from previous page

Grupo	test
GrupoD	z19
GrupoD	z20
GrupoD	z21
GrupoD	z22
GrupoD	z23
GrupoD	z24
GrupoD	z25
GrupoD	z27
GrupoD	z28
GrupoD	z29
GrupoD	z30
GrupoD	z31
GrupoD	z32
GrupoD	z33
GrupoD	z34
GrupoD	z35
GrupoD	z36
GrupoD	z37
GrupoD	z38
GrupoD	z39
GrupoD	z40
GrupoD	z41
GrupoD	z42
GrupoD	z43
GrupoD	z44
GrupoD	z45
GrupoD	z46
GrupoD	z48
GrupoD	z49
GrupoD	z50
GrupoD	z51
GrupoD	z53
GrupoD	z54
GrupoD	z55
GrupoD	z56
GrupoD	z58
GrupoD	z59
GrupoF	z02
GrupoF	z03
GrupoF	z05
GrupoF	z06
GrupoF	z07
GrupoF	z08
GrupoF	z09
GrupoF	z14
GrupoF	z15
GrupoF	z16
GrupoF	z17
GrupoF	z18
GrupoF	z19
GrupoF	z20
GrupoF	z21
GrupoF	z22
GrupoF	z23
GrupoF	z24
GrupoF	z25
GrupoF	z26
GrupoF	z27
GrupoF	z28
GrupoF	z29
GrupoF	z30
GrupoF	z31
GrupoF	z32
GrupoF	z33
GrupoF	z34
GrupoF	z35

Continued on next page

Continued from previous page

Grupo	test
GrupoF	z36
GrupoF	z37
GrupoF	z38
GrupoF	z39
GrupoF	z40
GrupoF	z41
GrupoF	z42
GrupoF	z43
GrupoF	z44
GrupoF	z45
GrupoF	z46
GrupoF	z48
GrupoF	z49
GrupoF	z50
GrupoF	z51
GrupoF	z54
GrupoF	z55
GrupoF	z56
GrupoF	z57
GrupoK	z00
GrupoK	z01
GrupoK	z02
GrupoK	z03
GrupoK	z06
GrupoK	z07
GrupoK	z08
GrupoK	z09
GrupoK	z18
GrupoK	z19
GrupoK	z21
GrupoK	z23
GrupoK	z24
GrupoK	z26
GrupoK	z35
GrupoK	z48
GrupoK	z49
GrupoK	z50
GrupoK	z51
GrupoK	z52
GrupoK	z56
GrupoK	z57
GrupoK	z58
GrupoK	z59
GrupoL	z00
GrupoL	z07
GrupoM	z00
GrupoN	z01
GrupoN	z02
GrupoN	z03
GrupoN	z04
GrupoN	z05
GrupoN	z06
GrupoN	z07
GrupoN	z08
GrupoN	z09
GrupoN	z10
GrupoN	z11
GrupoN	z12
GrupoN	z13
GrupoN	z14
GrupoN	z15
GrupoN	z16
GrupoN	z17
GrupoN	z18
GrupoN	z19
GrupoN	z20

Continued on next page

Continued from previous page

Grupo	test
GrupoN	z21
GrupoN	z22
GrupoN	z23
GrupoN	z24
GrupoN	z25
GrupoN	z26
GrupoN	z27
GrupoN	z28
GrupoN	z29
GrupoN	z30
GrupoN	z31
GrupoN	z32
GrupoN	z33
GrupoN	z34
GrupoN	z35
GrupoN	z36
GrupoN	z37
GrupoN	z38
GrupoN	z39
GrupoN	z40
GrupoN	z41
GrupoN	z42
GrupoN	z43
GrupoN	z44
GrupoN	z45
GrupoN	z46
GrupoN	z48
GrupoN	z49
GrupoN	z50
GrupoN	z51
GrupoN	z52
GrupoN	z53
GrupoN	z54
GrupoN	z55
GrupoN	z56
GrupoN	z57
GrupoN	z58
GrupoN	z59
GrupoR	z07
GrupoT	z07
GrupoZ	z02
GrupoZ	z03
GrupoZ	z06
GrupoZ	z07
GrupoZ	z08
GrupoZ	z21
GrupoZ	z23
GrupoZ	z24
GrupoZ	z35
GrupoZ	z48
GrupoZ	z49
GrupoZ	z50
GrupoZ	z51

2.3 Tabela completa

A tabela completa consiste em uma tabela com as seguintes colunas:

- Grupo: o identificador do grupo
- test: o identificador do teste (veja o `tgz` com os arquivos)
- TCNO: valor de retorno do programa que compara a árvore
- Output: saída do script que tenta executar o programa do grupo
- Decision: decisão final se o teste passou ou não.

```
read_csv("e3_output_objetivo_completa.csv", col_types=cols(), progress=FALSE)
```

```
# A tibble: 1,121 × 5
   Grupo  test   TCNO Output Decision
   <chr>  <chr> <dbl> <chr>  <lgl>
 1 GrupoA z00       0 <NA>   TRUE
 2 GrupoA z01       0 <NA>   TRUE
 3 GrupoA z02       0 <NA>   TRUE
 4 GrupoA z03       0 <NA>   TRUE
 5 GrupoA z04       0 <NA>   TRUE
 6 GrupoA z05       0 <NA>   TRUE
 7 GrupoA z06       0 <NA>   TRUE
 8 GrupoA z07       0 <NA>   TRUE
 9 GrupoA z08       0 <NA>   TRUE
10 GrupoA z09       0 <NA>   TRUE
#  1,111 more rows
#  Use `print(n = ...)` to see more rows
```

2.4 Tabela simplificada

A tabela simplificada consiste em uma tabela com 5 colunas. As colunas são:

- Grupo: o identificador do grupo
- test: o identificador do teste (veja o `tgz` com os arquivos)
- TCNO: valor de retorno do programa que compara a árvore
- Decision: decisão final se o teste passou ou não.

```
read_csv("e3_output_objetivo.csv", col_types=cols(), progress=FALSE)
```

```
# A tibble: 1,121 × 4
   Grupo  test   TCNO Decision
   <chr>  <chr> <dbl> <lgl>
 1 GrupoA z00       0 TRUE
 2 GrupoA z01       0 TRUE
 3 GrupoA z02       0 TRUE
 4 GrupoA z03       0 TRUE
 5 GrupoA z04       0 TRUE
 6 GrupoA z05       0 TRUE
 7 GrupoA z06       0 TRUE
 8 GrupoA z07       0 TRUE
 9 GrupoA z08       0 TRUE
10 GrupoA z09       0 TRUE
#  1,111 more rows
#  Use `print(n = ...)` to see more rows
```

2.5 Tabela com saída

Esta tabela mostra a saída, normalmente indica o erro.

```
read_csv("e3_output_errors.csv", col_types=cols(), progress=FALSE)
```

```
# A tibble: 58 × 3
   Grupo  test  Output
   <chr>  <chr> <chr>
 1 GrupoN z01   Segmentation fault
 2 GrupoN z02   Segmentation fault
 3 GrupoN z03   Segmentation fault
 4 GrupoN z04   Segmentation fault
 5 GrupoN z05   Segmentation fault
 6 GrupoN z06   Segmentation fault
 7 GrupoN z07   Segmentation fault
 8 GrupoN z08   Segmentation fault
 9 GrupoN z09   Segmentation fault
10 GrupoN z10   Segmentation fault
#  48 more rows
#  Use `print(n = ...)` to see more rows
```

3 Pesos

Grupo	E3.P
GrupoA	1
GrupoB	1
GrupoC	1
GrupoD	1
GrupoE	1
GrupoF	1
GrupoG	1
GrupoH	1
GrupoJ	1
GrupoK	1
GrupoL	1
GrupoM	1
GrupoN	1
GrupoO	1
GrupoP	1
GrupoR	1
GrupoS	1
GrupoT	1
GrupoZ	1

4 Final

Grupo	Etapa	E3.O	E3.S	E3.P	E3
GrupoA	E3	10	10	1	10
GrupoG	E3	10	10	1	10
GrupoS	E3	10	10	1	10
GrupoE	E3	10	9.675	1	9.84
GrupoO	E3	10	9.675	1	9.84
GrupoR	E3	9.83	9.675	1	9.75
GrupoT	E3	9.83	9.675	1	9.75
GrupoH	E3	10	9.35	1	9.68
GrupoP	E3	10	9.35	1	9.68
GrupoC	E3	9.66	9.675	1	9.67
GrupoL	E3	9.66	9.675	1	9.67
GrupoM	E3	9.83	9.35	1	9.59
GrupoJ	E3	10	8.7	1	9.35
GrupoB	E3	9.32	9.35	1	9.34
GrupoZ	E3	7.8	8.7	1	8.25
GrupoK	E3	5.93	8.05	1	6.99
GrupoF	E3	1.86	4.8	1	3.33
GrupoD	E3	0.68	3.5	1	2.09
GrupoN	E3	0.17	2.5	1	1.33

5 Recuperação

Grupos em recuperação da E3:

- Em Moodle já configurado para tal, use o novo link "Recuperação E3"
- Política de recuperação ativada (vejam regras gerais)

Grupo	Entrega.Em.Atraso	Recuperação
GrupoD	FALSE	TRUE
GrupoF	FALSE	TRUE
GrupoN	FALSE	TRUE