COMP LabBook 2025 1 - E4

Lucas M. Schnorr

November 10, 2025

Contents

1 Introdução

- Arquivo `e4_make.log` contém o log de compilação
- Arquivo `e4.log` contém o log de execução dos testes
- Arquivo `e4_arquivo.log` contém também o log de execução dos testes, mas simplificado para transformação em formato tabular e contagem das respostas.
- Arquivo `e4_error_code.csv` contém o mapeamento do nome para o código de erro, de maneira a intepretar os nomes dos erros nos arquivos de testes
- Arquivo `e4_expected_code.csv` contém o código esperado para cada um dos testes, que pode ser o nome quando trata-se de um erro semântico, ou 0 quando não existe erro esperado.
- Arquivo `e4_objetivo.csv` fornece um sumário da nota objetiva.
  - A coluna **N** indica a quantidade de testes, a coluna **Correto** indica a quantidade de testes correto, e a coluna **E4.O** a nota objetiva final, que é oriunda de Correto/N.
- Arquivo `e4_tests_with_expected.csv` fornece a decisão individual de cada teste. A Coluna **Test** indica o teste utilizado (nome do arquivo), a coluna **Res** indica a resposta fornecida pelo programa do grupo (obtida a partir de $? após executar o programa em um script bash), a coluna **Output** indica o conteúdo da saída padrão fornecida pelo programa (quando existem múltiplas linhas, apenas a primeira é mostrada), a coluna **Code** indica o código de erro esperado para o teste, e, por fim, a coluna **TEST.WORKED** indica se o teste passou (TRUE) ou não (FALSE).

Para os testes que causam falha de segmentação, use o **gdb** para identificar o porquê após compilar seu programa com a opção -g para o compilador.

De uma maneira geral, veja o Relatório de erros abaixo procurando pelo seu identificador de grupo. Em seguida, temos o Histórico de comentários com algumas anotações durante a revisão de código, testes manuais, etc.

2 Relatório de erros

2.1 Falhas de segmentação e Liberação dupla (*double free*)

As seguintes entradas (veja coluna `Test`) causam falha de segmentação (Res = 139) ou liberação dupla de memória (Res = 134) no programa. Para estes testes, sugiro que o grupo compile os fontes com -g além do -c para em seguida executar o valgrind com `--leak-check=full` conforme anteriormente explicado. Isso permitirá ao grupo identificar a origem do problema, resolvendo-o.

Grupo	Test	Res	Code	TEST.WORKED
GrupoK	qwe06	139	0	FALSE
GrupoK	qwe12	139	10	FALSE

2.2 Erros não são reconhecidos pelo programa

Os erros semânticos não reconhecidos abaixo acontecem quando o código do programa retorna zero, quando na realidade se esperava um erro semântico, com o código que aparece na tabela. Para facilitar a compreensão de onde o equívoco acontece, lista-se na coluna `Test` a entrada de teste utilizada para identificar o problema.

Grupo	Test	Res	Code	Erro
GrupoA	qwe24	0	11	ERR_DECLARED
GrupoA	qwe48	0	42	ERR_WRONG_TYPE_ARGS
GrupoA	qwe69	0	30	ERR_WRONG_TYPE
GrupoB	qwe38	0	21	ERR_FUNCTION
GrupoB	qwe39	0	21	ERR_FUNCTION
GrupoB	qwe65	0	30	ERR_WRONG_TYPE
GrupoB	qwe66	0	30	ERR_WRONG_TYPE
GrupoB	qwe67	0	30	ERR_WRONG_TYPE
GrupoB	qwe68	0	30	ERR_WRONG_TYPE
GrupoB	qwe69	0	30	ERR_WRONG_TYPE
GrupoC	qwe23	0	11	ERR_DECLARED
GrupoD	qwe68	0	30	ERR_WRONG_TYPE
GrupoD	qwe78	0	30	ERR_WRONG_TYPE
GrupoD	qwe79	0	30	ERR_WRONG_TYPE
GrupoD	qwe80	0	30	ERR_WRONG_TYPE
GrupoD	qwe81	0	30	ERR_WRONG_TYPE
GrupoE	qwe24	0	11	ERR_DECLARED
GrupoE	qwe65	0	30	ERR_WRONG_TYPE
GrupoE	qwe66	0	30	ERR_WRONG_TYPE
GrupoE	qwe67	0	30	ERR_WRONG_TYPE
GrupoG	qwe00	0	10	ERR_UNDECLARED
GrupoG	qwe01	0	10	ERR_UNDECLARED
GrupoG	qwe02	0	10	ERR_UNDECLARED
GrupoG	qwe03	0	10	ERR_UNDECLARED
GrupoG	qwe10	0	10	ERR_UNDECLARED
GrupoG	qwe11	0	10	ERR_UNDECLARED
GrupoG	qwe12	0	10	ERR_UNDECLARED
GrupoG	qwe13	0	10	ERR_UNDECLARED
GrupoG	qwe14	0	10	ERR_UNDECLARED
GrupoG	qwe15	0	10	ERR_UNDECLARED
GrupoG	qwe16	0	11	ERR_DECLARED
GrupoG	qwe17	0	11	ERR_DECLARED
GrupoG	qwe18	0	11	ERR_DECLARED
GrupoG	qwe19	0	11	ERR_DECLARED
GrupoG	qwe20	0	11	ERR_DECLARED
GrupoG	qwe21	0	11	ERR_DECLARED
GrupoG	qwe22	0	11	ERR_DECLARED
GrupoG	qwe23	0	11	ERR_DECLARED
GrupoG	qwe24	0	11	ERR_DECLARED
GrupoG	qwe27	0	11	ERR_DECLARED
GrupoG	qwe28	0	11	ERR_DECLARED
GrupoG	qwe29	0	11	ERR_DECLARED
GrupoG	qwe34	0	20	ERR_VARIABLE
GrupoG	qwe35	0	20	ERR_VARIABLE
GrupoG	qwe36	0	20	ERR_VARIABLE
GrupoG	qwe37	0	20	ERR_VARIABLE
GrupoG	qwe38	0	21	ERR_FUNCTION
GrupoG	qwe39	0	21	ERR_FUNCTION
GrupoG	qwe40	0	21	ERR_FUNCTION
GrupoG	qwe41	0	21	ERR_FUNCTION
GrupoG	qwe42	0	40	ERR_MISSING_ARGS
GrupoG	qwe43	0	40	ERR_MISSING_ARGS
GrupoG	qwe44	0	41	ERR_EXCESS_ARGS
GrupoG	qwe45	0	41	ERR_EXCESS_ARGS
GrupoG	qwe46	0	42	ERR_WRONG_TYPE_ARGS
GrupoG	qwe47	0	42	ERR_WRONG_TYPE_ARGS
GrupoG	qwe48	0	42	ERR_WRONG_TYPE_ARGS
GrupoG	qwe49	0	42	ERR_WRONG_TYPE_ARGS
GrupoG	qwe50	0	42	ERR_WRONG_TYPE_ARGS
GrupoG	qwe51	0	42	ERR_WRONG_TYPE_ARGS
GrupoG	qwe52	0	42	ERR_WRONG_TYPE_ARGS
GrupoG	qwe53	0	42	ERR_WRONG_TYPE_ARGS
GrupoG	qwe54	0	42	ERR_WRONG_TYPE_ARGS
GrupoG	qwe55	0	42	ERR_WRONG_TYPE_ARGS
GrupoG	qwe56	0	42	ERR_WRONG_TYPE_ARGS

Continued on next page

Continued from previous page

Grupo	Test	Res	Code	Erro
GrupoG	qwe57	0	42	ERR_WRONG_TYPE_ARGS
GrupoG	qwe58	0	42	ERR_WRONG_TYPE_ARGS
GrupoG	qwe59	0	42	ERR_WRONG_TYPE_ARGS
GrupoG	qwe60	0	42	ERR_WRONG_TYPE_ARGS
GrupoG	qwe61	0	42	ERR_WRONG_TYPE_ARGS
GrupoG	qwe62	0	42	ERR_WRONG_TYPE_ARGS
GrupoG	qwe63	0	42	ERR_WRONG_TYPE_ARGS
GrupoG	qwe64	0	42	ERR_WRONG_TYPE_ARGS
GrupoG	qwe65	0	30	ERR_WRONG_TYPE
GrupoG	qwe66	0	30	ERR_WRONG_TYPE
GrupoG	qwe67	0	30	ERR_WRONG_TYPE
GrupoG	qwe68	0	30	ERR_WRONG_TYPE
GrupoG	qwe69	0	30	ERR_WRONG_TYPE
GrupoG	qwe70	0	30	ERR_WRONG_TYPE
GrupoG	qwe71	0	30	ERR_WRONG_TYPE
GrupoG	qwe72	0	30	ERR_WRONG_TYPE
GrupoG	qwe73	0	30	ERR_WRONG_TYPE
GrupoG	qwe74	0	30	ERR_WRONG_TYPE
GrupoG	qwe75	0	30	ERR_WRONG_TYPE
GrupoG	qwe76	0	30	ERR_WRONG_TYPE
GrupoG	qwe77	0	30	ERR_WRONG_TYPE
GrupoG	qwe78	0	30	ERR_WRONG_TYPE
GrupoG	qwe79	0	30	ERR_WRONG_TYPE
GrupoG	qwe80	0	30	ERR_WRONG_TYPE
GrupoG	qwe81	0	30	ERR_WRONG_TYPE
GrupoG	qwe82	0	30	ERR_WRONG_TYPE
GrupoI	qwe24	0	11	ERR_DECLARED
GrupoI	qwe81	0	30	ERR_WRONG_TYPE
GrupoI	qwe82	0	30	ERR_WRONG_TYPE
GrupoJ	qwe10	0	10	ERR_UNDECLARED
GrupoJ	qwe14	0	10	ERR_UNDECLARED
GrupoJ	qwe15	0	10	ERR_UNDECLARED
GrupoJ	qwe40	0	21	ERR_FUNCTION
GrupoJ	qwe41	0	21	ERR_FUNCTION
GrupoJ	qwe81	0	30	ERR_WRONG_TYPE
GrupoK	qwe23	0	11	ERR_DECLARED
GrupoK	qwe24	0	11	ERR_DECLARED
GrupoK	qwe69	0	30	ERR_WRONG_TYPE
GrupoL	qwe65	0	30	ERR_WRONG_TYPE
GrupoL	qwe66	0	30	ERR_WRONG_TYPE
GrupoL	qwe67	0	30	ERR_WRONG_TYPE
GrupoL	qwe68	0	30	ERR_WRONG_TYPE
GrupoL	qwe69	0	30	ERR_WRONG_TYPE
GrupoO	qwe23	0	11	ERR_DECLARED
GrupoO	qwe24	0	11	ERR_DECLARED
GrupoP	qwe16	0	11	ERR_DECLARED
GrupoP	qwe24	0	11	ERR_DECLARED
GrupoP	qwe27	0	11	ERR_DECLARED
GrupoP	qwe28	0	11	ERR_DECLARED
GrupoP	qwe29	0	11	ERR_DECLARED
GrupoP	qwe69	0	30	ERR_WRONG_TYPE
GrupoQ	qwe24	0	11	ERR_DECLARED
GrupoQ	qwe48	0	42	ERR_WRONG_TYPE_ARGS
GrupoR	qwe24	0	11	ERR_DECLARED
GrupoT	qwe23	0	11	ERR_DECLARED
GrupoV	qwe10	0	10	ERR_UNDECLARED
GrupoV	qwe14	0	10	ERR_UNDECLARED
GrupoV	qwe15	0	10	ERR_UNDECLARED
GrupoV	qwe78	0	30	ERR_WRONG_TYPE
GrupoV	qwe79	0	30	ERR_WRONG_TYPE
GrupoV	qwe80	0	30	ERR_WRONG_TYPE
GrupoV	qwe81	0	30	ERR_WRONG_TYPE
GrupoV	qwe82	0	30	ERR_WRONG_TYPE
GrupoW	qwe24	0	11	ERR_DECLARED
GrupoW	qwe69	0	30	ERR_WRONG_TYPE
GrupoW	qwe81	0	30	ERR_WRONG_TYPE

Continued on next page

Continued from previous page

Grupo	Test	Res	Code	Erro
GrupoW	qwe82	0	30	ERR_WRONG_TYPE
GrupoZ	qwe69	0	30	ERR_WRONG_TYPE
GrupoZ	qwe81	0	30	ERR_WRONG_TYPE
GrupoZ	qwe82	0	30	ERR_WRONG_TYPE

2.3 Detecta erro sintático para entradas sintaticamente válidas

Para as entradas listadas nesta tabela, o programa detecta um erro sintático (código de retorno 1 na coluna `Res`) para entradas que são consideradas sintaticamente válidas de acordo com o definido nas etapas anteriores. Para fins de indicação, lista-se na coluna `Code` o erro *semântico* que deveria ter sido lançado para o teste.

Grupo	Test	Res	Code	TEST.WORKED
GrupoC	qwe06	1	0	FALSE
GrupoF	qwe66	1	30	FALSE
GrupoF	qwe78	1	30	FALSE
GrupoF	qwe79	1	30	FALSE
GrupoF	qwe80	1	30	FALSE
GrupoF	qwe81	1	30	FALSE
GrupoH	qwe04	1	0	FALSE
GrupoH	qwe05	1	0	FALSE
GrupoH	qwe06	1	0	FALSE
GrupoH	qwe07	1	0	FALSE
GrupoH	qwe08	1	0	FALSE
GrupoH	qwe09	1	0	FALSE
GrupoH	qwe23	1	11	FALSE
GrupoH	qwe24	1	11	FALSE
GrupoH	qwe25	1	0	FALSE
GrupoH	qwe29	1	11	FALSE
GrupoH	qwe30	1	0	FALSE
GrupoH	qwe31	1	0	FALSE
GrupoH	qwe32	1	0	FALSE
GrupoH	qwe33	1	0	FALSE
GrupoH	qwe41	1	21	FALSE
GrupoH	qwe42	1	40	FALSE
GrupoH	qwe43	1	40	FALSE
GrupoH	qwe46	1	42	FALSE
GrupoH	qwe47	1	42	FALSE
GrupoH	qwe48	1	42	FALSE
GrupoH	qwe49	1	42	FALSE
GrupoH	qwe50	1	42	FALSE
GrupoH	qwe51	1	42	FALSE
GrupoH	qwe52	1	42	FALSE
GrupoH	qwe53	1	42	FALSE
GrupoH	qwe54	1	42	FALSE
GrupoH	qwe55	1	42	FALSE
GrupoH	qwe56	1	42	FALSE
GrupoH	qwe57	1	42	FALSE
GrupoH	qwe58	1	42	FALSE
GrupoH	qwe59	1	42	FALSE
GrupoH	qwe60	1	42	FALSE
GrupoH	qwe61	1	42	FALSE
GrupoH	qwe62	1	42	FALSE
GrupoH	qwe63	1	42	FALSE
GrupoH	qwe64	1	42	FALSE
GrupoH	qwe69	1	30	FALSE
GrupoH	qwe78	1	30	FALSE
GrupoH	qwe79	1	30	FALSE
GrupoH	qwe80	1	30	FALSE
GrupoH	qwe81	1	30	FALSE
GrupoH	qwe82	1	30	FALSE
GrupoJ	qwe03	1	10	FALSE
GrupoJ	qwe06	1	0	FALSE
GrupoJ	qwe11	1	10	FALSE
GrupoJ	qwe12	1	10	FALSE

Continued on next page

Continued from previous page

Grupo	Test	Res	Code	TEST.WORKED
GrupoJ	qwe13	1	10	FALSE
GrupoJ	qwe78	1	30	FALSE
GrupoJ	qwe79	1	30	FALSE
GrupoJ	qwe80	1	30	FALSE
GrupoL	qwe06	1	0	FALSE
GrupoL	qwe12	1	10	FALSE

2.4 Detecta erro semântico em situações sem erro semântico esperado

O programa detecta erro semântico em testes onde nenhum erro semântico é esperado. Somente execuções que vão até o final sem falha de segmentação ou semelhantes são listadas.

Grupo	Test	Res	Code	TEST.WORKED
GrupoB	qwe25	11	0	FALSE
GrupoB	qwe26	11	0	FALSE
GrupoC	qwe26	11	0	FALSE
GrupoD	qwe31	10	0	FALSE
GrupoD	qwe33	10	0	FALSE
GrupoI	qwe33	41	0	FALSE
GrupoK	qwe31	10	0	FALSE
GrupoK	qwe33	10	0	FALSE
GrupoL	qwe25	11	0	FALSE
GrupoL	qwe26	11	0	FALSE
GrupoM	qwe26	11	0	FALSE
GrupoO	qwe33	10	0	FALSE
GrupoP	qwe04	10	0	FALSE
GrupoP	qwe05	10	0	FALSE
GrupoP	qwe06	10	0	FALSE
GrupoP	qwe07	10	0	FALSE
GrupoP	qwe08	10	0	FALSE
GrupoP	qwe09	10	0	FALSE
GrupoP	qwe25	11	0	FALSE
GrupoP	qwe30	10	0	FALSE
GrupoP	qwe32	10	0	FALSE
GrupoP	qwe33	41	0	FALSE
GrupoR	qwe33	41	0	FALSE
GrupoS	qwe25	11	0	FALSE
GrupoS	qwe26	11	0	FALSE

2.5 Detecta o erro semântico errado

O programa detecta um erro semântico diferente daquele esperado. Na tabela apresenta-se na coluna `Test` o identificador do teste onde o programa do grupo gera um código de erro (coluna `Res`) quando o esperado é o especificado na coluna `Code`. Somente execuções que vão até o final sem falha de segmentação ou semelhantes são listadas.

Grupo	Test	Res	Code	TEST.WORKED
GrupoA	qwe54	41	42	FALSE
GrupoA	qwe55	41	42	FALSE
GrupoA	qwe56	41	42	FALSE
GrupoA	qwe61	41	42	FALSE
GrupoA	qwe62	41	42	FALSE
GrupoB	qwe42	41	40	FALSE
GrupoB	qwe43	41	40	FALSE
GrupoB	qwe49	41	42	FALSE
GrupoB	qwe50	41	42	FALSE
GrupoB	qwe51	41	42	FALSE
GrupoB	qwe52	41	42	FALSE
GrupoB	qwe53	41	42	FALSE
GrupoB	qwe54	41	42	FALSE
GrupoB	qwe55	41	42	FALSE
GrupoB	qwe56	41	42	FALSE
GrupoB	qwe57	41	42	FALSE
GrupoB	qwe58	41	42	FALSE

Continued on next page

Continued from previous page

Grupo	Test	Res	Code	TEST.WORKED
GrupoB	qwe59	41	42	FALSE
GrupoB	qwe60	41	42	FALSE
GrupoB	qwe61	41	42	FALSE
GrupoB	qwe62	41	42	FALSE
GrupoB	qwe63	41	42	FALSE
GrupoB	qwe64	41	42	FALSE
GrupoD	qwe38	10	21	FALSE
GrupoD	qwe44	10	41	FALSE
GrupoD	qwe45	10	41	FALSE
GrupoD	qwe82	10	30	FALSE
GrupoJ	qwe49	41	42	FALSE
GrupoJ	qwe50	41	42	FALSE
GrupoJ	qwe51	41	42	FALSE
GrupoJ	qwe52	41	42	FALSE
GrupoJ	qwe53	41	42	FALSE
GrupoJ	qwe54	41	42	FALSE
GrupoJ	qwe55	41	42	FALSE
GrupoJ	qwe56	41	42	FALSE
GrupoJ	qwe57	41	42	FALSE
GrupoJ	qwe58	41	42	FALSE
GrupoJ	qwe59	41	42	FALSE
GrupoJ	qwe60	41	42	FALSE
GrupoJ	qwe61	41	42	FALSE
GrupoJ	qwe62	41	42	FALSE
GrupoJ	qwe63	41	42	FALSE
GrupoJ	qwe64	41	42	FALSE
GrupoK	qwe38	10	21	FALSE
GrupoK	qwe39	20	21	FALSE
GrupoK	qwe44	10	41	FALSE
GrupoK	qwe45	10	41	FALSE
GrupoK	qwe82	10	30	FALSE
GrupoL	qwe41	11	21	FALSE
GrupoM	qwe34	21	20	FALSE
GrupoM	qwe35	21	20	FALSE
GrupoM	qwe36	21	20	FALSE
GrupoM	qwe37	21	20	FALSE
GrupoM	qwe38	20	21	FALSE
GrupoM	qwe39	20	21	FALSE
GrupoM	qwe40	20	21	FALSE
GrupoM	qwe41	20	21	FALSE
GrupoM	qwe49	41	42	FALSE
GrupoM	qwe50	41	42	FALSE
GrupoM	qwe51	41	42	FALSE
GrupoM	qwe52	41	42	FALSE
GrupoM	qwe53	41	42	FALSE
GrupoM	qwe54	41	42	FALSE
GrupoM	qwe55	41	42	FALSE
GrupoM	qwe56	41	42	FALSE
GrupoM	qwe57	41	42	FALSE
GrupoM	qwe58	41	42	FALSE
GrupoM	qwe59	41	42	FALSE
GrupoM	qwe60	41	42	FALSE
GrupoM	qwe61	41	42	FALSE
GrupoM	qwe62	41	42	FALSE
GrupoM	qwe63	41	42	FALSE
GrupoM	qwe64	41	42	FALSE
GrupoP	qwe34	10	20	FALSE
GrupoP	qwe36	10	20	FALSE
GrupoP	qwe39	10	21	FALSE
GrupoP	qwe40	10	21	FALSE
GrupoP	qwe41	10	21	FALSE
GrupoP	qwe42	10	40	FALSE
GrupoP	qwe43	10	40	FALSE
GrupoP	qwe46	10	42	FALSE
GrupoP	qwe47	10	42	FALSE
GrupoP	qwe48	10	42	FALSE

Continued on next page

Continued from previous page

Grupo	Test	Res	Code	TEST.WORKED
GrupoP	qwe49	10	42	FALSE
GrupoP	qwe50	10	42	FALSE
GrupoP	qwe51	10	42	FALSE
GrupoP	qwe52	10	42	FALSE
GrupoP	qwe53	10	42	FALSE
GrupoP	qwe54	10	42	FALSE
GrupoP	qwe55	10	42	FALSE
GrupoP	qwe56	10	42	FALSE
GrupoP	qwe57	10	42	FALSE
GrupoP	qwe58	10	42	FALSE
GrupoP	qwe59	10	42	FALSE
GrupoP	qwe60	10	42	FALSE
GrupoP	qwe61	10	42	FALSE
GrupoP	qwe62	10	42	FALSE
GrupoP	qwe63	10	42	FALSE
GrupoP	qwe64	10	42	FALSE
GrupoP	qwe82	10	30	FALSE
GrupoV	qwe49	41	42	FALSE
GrupoV	qwe50	41	42	FALSE
GrupoV	qwe51	41	42	FALSE
GrupoV	qwe52	41	42	FALSE
GrupoV	qwe53	41	42	FALSE
GrupoV	qwe54	41	42	FALSE
GrupoV	qwe55	41	42	FALSE
GrupoV	qwe56	41	42	FALSE
GrupoV	qwe57	41	42	FALSE
GrupoV	qwe58	41	42	FALSE
GrupoV	qwe59	41	42	FALSE
GrupoV	qwe60	41	42	FALSE
GrupoV	qwe61	41	42	FALSE
GrupoV	qwe62	41	42	FALSE
GrupoV	qwe63	41	42	FALSE
GrupoV	qwe64	41	42	FALSE

3 Histórico de comentários

3.1 Introdução

Os comentários abaixos focam na observação do código e a experiência do professor da execução dos testes com o código do grupo. Os comentários abaixo não mencionam explicitamente os erros detectados através dos testes automáticos. Para estes, a sugestão é olhar o ZIP e o TGZ e a introdução deste relatório para decodificar os ajustes que devem ser feitos. Caso houverem dúvidas, entre em contato.

3.2 GrupoA

3.2.1 E4

- O tipo de dado pode ser int ou float, não tem como ser `SEMANTIC_TYPE_UNDEFINED`
- Sobre mensagens de erros
  - De uma maneira geral, faltou mencionar os nomes dos identificadores envolvidos em erros semânticos
- Flag -fsanitize=address não presente no log de compilação
- 27 testes acusam problemas de alocação de memória

3.2.2 E3

☒ Conforme explicitado na E3, o comando na avaliação objetiva é

```
./etapa3 < entrada > saida.dot
```

Antes de qualquer coisa, isso quer dizer que o programa `etapa3` deve realizar o registro da saída na "saída padrão", e não no arquivo `saida.dot`. O comando redireciona para o arquivo `saida.dot` neste exemplo, mas poderia ser qualquer outro nome de arquivo, inclusive poderia não ser um arquivo, mas outro programa. A solução do grupo sempre registra a saída no arquivo `saida.dot`, em desacordo portanto com a especificação.

☒ Fatorar código no arquivo `scanner.l` pois existem trechos muito similares (para não dizer idênticos) em nas ações associadas aos tokens que geram valor.

- O `-fsanitize=address` não é empregado na hora de compilação com `make`
- Fatorar o código das ações no `parser.y` em situações onde é possível

3.2.3 E2

☒ Evitar de enviar binários compilados

  - Fazer `make clean` antes de `make tgz`

☒ Dar uma organizada no Makefile

3.2.4 E1

☒ Comentar o código.

☒ Todos os arquivos devem conter a identificação do grupo e os seus membros

☒ Comandos como `rm`, quando chamados no Makefile, podem vir acompanhados do parâmetro `-f` (force)

☒ Adicionar um alvo `clean` que remove arquivos gerados/compilados

☒ Empregar `-Wall` e `-Werror`

☒ Empregar `-fsanitize=address`

3.3 GrupoB

3.3.1 E4

- Chave da entrada na tabela não é o lexema (int chave)
- O tipo de dado está registrado em uma variável `char`, poderia ter sido um `enum`, com `typedef` para tornar o código mais legível.

3.3.2 E3

☒ Fatorar código no arquivo `scanner.l` pois existem trechos muito similares (para não dizer idênticos) em nas ações associadas aos tokens que geram valor.

- Fatorar o código das ações no `parser.y` em situações onde é possível

3.3.3 E2

☒ Precedência em expressões não foi implementada corretamente

  - Ainda que tenha essa separação com comentários

3.3.4 E1

☒ Usar wildcards no Makefile (*it makes your life easier*)

☒ Todos os arquivos devem conter a identificação do grupo e os seus membros

☒ Comandos como `rm`, quando chamados no Makefile, podem vir acompanhados do parâmetro `-f` (force)

☒ Adicionar um alvo `clean` que remove arquivos gerados/compilados

☒ Empregar `-Wall` e `-Werror`

☒ Empregar `-fsanitize=address`

3.4 GrupoC

3.4.1 E4

- O tipo de dado está registrado em uma variável `int`, poderia ter sido um `enum`, com `typedef` para tornar o código mais legível.
- 1 teste acusa problema de alocação de memória

3.4.2 E3

☒ Fatorar código no arquivo `scanner.l` pois existem trechos muito similares (para não dizer idênticos) em nas ações associadas aos tokens que geram valor.

- Fatorar o código das ações no `parser.y` em situações onde é possível

3.4.3 E2

- Podem usar `%option noinput nounput` no scanner.l ao invés daquelas flags de compilação para contornar o aviso de compilação

3.4.4 E1

☒ O arquivo era para ser chamado `scanner.l`, lembra?

☒ Usar wildcards no Makefile (*it makes your life easier*)

☒ Todos os arquivos devem conter a identificação do grupo e os seus membros

☒ Comandos como `rm`, quando chamados no Makefile, podem vir acompanhados do parâmetro `-f` (force)

☒ Adicionar um alvo `clean` que remove arquivos gerados/compilados

- Empregar `-Wall` e `-Werror`

☒ Empregar `-fsanitize=address`

3.5 GrupoD

3.5.1 E4

- Chave da tabela é valor léxico, e não char*
- O tipo de dado pode ser int ou float, não void.
- Sobre mensagens de erros
  - De uma maneira geral, faltou mencionar os nomes dos identificadores envolvidos em erros semânticos

3.5.2 E3

☒ Fatorar código no arquivo `scanner.l` pois existem trechos muito similares (para não dizer idênticos) em nas ações associadas aos tokens que geram valor.

- Fatorar o código das ações no `parser.y` em situações onde é possível

3.5.3 E2

- Podem usar `%option noinput nounput` no scanner.l ao invés daquelas flags de compilação para contornar o aviso de compilação

3.5.4 E1

☒ Comentar mais o código.

☒ Todos os arquivos devem conter a identificação do grupo e os seus membros

☒ Comandos como `rm`, quando chamados no Makefile, podem vir acompanhados do parâmetro `-f` (force)

☒ Adicionar um alvo `clean` que remove arquivos gerados/compilados

☒ Empregar `-Wall` e `-Werror`

☒ Empregar `-fsanitize=address`

3.6 GrupoE (o grupo mais gaudério)

3.6.1 E4

- O tipo de dado está registrado em uma variável `int`, poderia ter sido um `enum`, com `typedef` para tornar o código mais legível.
- Gostei do "Bah tchê!", "Mas guri!", "Mas que barbaridade!", "Bah, mas ta tri errado!", todos!
- Sobre mensagens de erros
  - De uma maneira geral, faltou mencionar os nomes dos identificadores envolvidos em erros semânticos

3.6.2 E3

☒ O `-fsanitize=address` não é empregado na hora de compilação com `make`

  - Pelo menos quando se executa `make`, não aparece uma linha que inclua gcc, -fsanitize=address, e parser.tab.c. Talvez algo obscuro no Makefile.

- Fatorar o código das ações no `parser.y` em situações onde é possível

3.6.3 E2

☒ O que aquele `%union` está fazendo ali?

  - O que aqueles `<tipo>` estão fazendo ali?

3.6.4 E1

☒ Comentar mais o código.

☒ Usar wildcards no Makefile (*it makes your life easier*)

- Todos os arquivos devem conter a identificação do grupo e os seus membros

☒ Comandos como `rm`, quando chamados no Makefile, podem vir acompanhados do parâmetro `-f` (force)

☒ Adicionar um alvo `clean` que remove arquivos gerados/compilados

☒ Empregar `-Wall` e `-Werror`

- Empregar `-fsanitize=address`

3.7 GrupoF

3.7.1 E4

- Entrega em atraso.
- Chave é (possivelmente) `lex_value`, e não char*
- 5 testes acusam problemas de alocação de memória

3.7.2 E3

- Fatorar o código das ações no `parser.y` em situações onde é possível

3.7.3 E2

☒ Temos conflitos!

  - `parser.y: warning: 5 shift/reduce conflicts [-Wconflicts-sr]`

3.7.4 E1

☒ Comentar o código.

☒ Usar wildcards no Makefile (*it makes your life easier*)

- Todos os arquivos devem conter a identificação do grupo e os seus membros

☒ Comandos como `rm`, quando chamados no Makefile, podem vir acompanhados do parâmetro `-f` (force)

☒ Adicionar um alvo `clean` que remove arquivos gerados/compilados

- Empregar `-Wall` e `-Werror`

☒ Empregar `-fsanitize=address`

3.8 GrupoG

3.8.1 E4

- Nenhum erro semântico é detectado

3.8.2 E3

☒ Todos os testes falham pois ha um link gerado no parser.y linha 137, que por sua vez chama asd.c linha 11 e asd.c linha 9. Isso ocorre porque nesta linha o grupo decidiu criar nós na AST que estão fora da especificação da E3 conforme comentário do grupo "Representa os tipos da linguagem, criando nós simples para eles.". Em nenhum lugar da especificação consta geração de nós para tipos.

☒ Realizando a alteração, obtemos

```
tipoVar: TK_INTEIRO { $$ = NULL; } | TK_DECIMAL { $$ = NULL; };
```

mas isso causa implicações visto que por vezes tipoVar estava sendo pendurado na árvore erroneamente. Por exemplo, nas regras de `declaracaoVariavel` (que por sua vez nem deveria ir para a árvore), conforme especificação da E3. Ainda por exemplo, `declaracaoVariavelComandoSimples` (que por sua vez nem deveria ir para a árvore), conforme especificação da E3.

☒ Fatorar código no arquivo `scanner.l` pois existem trechos muito similares (para não dizer idênticos) em nas ações associadas aos tokens que geram valor.

- Fatorar o código das ações no `parser.y` em situações onde é possível

3.8.3 E2

3.8.4 E1

☒ Remover os arquivos ocultos do mac

  – E não versioná-los no projeto

☒ Usar wildcards no Makefile (*it makes your life easier*)

☒ Todos os arquivos devem conter a identificação do grupo e os seus membros

  – Faltou no Makefile

☒ Comandos como `rm`, quando chamados no Makefile, podem vir acompanhados do parâmetro `-f` (force)

☒ Adicionar um alvo `clean` que remove arquivos gerados/compilados

☒ Empregar `-Wall` e `-Werror`

☒ Empregar `-fsanitize=address`

3.9 GrupoH

3.9.1 E4

- O tipo de dado pode ser int ou float, não tem como ser `TYPE_UNKNOWN`
- 29 testes acusam problemas de alocação de memória

3.9.2 E3

- Fatorar o código das ações no `parser.y` em situações onde é possível

3.9.3 E2

☒ Colocar nome em todos os arquivos (parser.y, makefile)

3.9.4 E1

☒ A regra da linha 46 nunca fará match, pois há uma regra que a precede que já captura espaço. Isso gera um warning. É importante remover todos os warnings.

☒ Estes comandos precisaram ser executados para resolver a situação

```
mv -v etapa1/* .
rm -rf etapa1
```

☐ Remover todos os arquivos ocultos antes de submeter a solução

  – Arquivos ocultos são aqueles que tipicamente começam por ponto

☒ Empregar um Makefile adequado, com wildcards

  – Existe redundância entre as regras do Makefile

☒ Todos os arquivos devem conter a identificação do grupo e os seus membros

☒ Comandos como `rm`, quando chamados no Makefile, podem vir acompanhados do parâmetro `-f` (force)

☒ Adicionar um alvo `clean` que remove arquivos gerados/compilados

- Empregar `-Wall` e `-Werror`

☒ Empregar `-fsanitize=address`

3.10 GrupoI

3.10.1 E4

- O tipo de dado está registrado em uma variável `int`, poderia ter sido um `enum`, com `typedef` para tornar o código mais legível.
- input e yyunput definidas mas não usadas, usar a opção do bison para evitar a definição

3.10.2 E3

☒ Vários conflitos de empilha/reduz e um de reduz/reduz

- Fatorar código no arquivo `scanner.l` pois existem trechos muito similares (para não dizer idênticos) em nas ações associadas aos tokens que geram valor.
- Fatorar o código das ações no `parser.y` em situações onde é possível

3.10.3 E2

☒ O código não estava na raiz do `tgz`

3.10.4 E1

☒ Comentar o código.

- Todos os arquivos devem conter a identificação do grupo e os seus membros

☒ Comandos como `rm`, quando chamados no Makefile, podem vir acompanhados do parâmetro `-f` (force)

☒ Adicionar um alvo `clean` que remove arquivos gerados/compilados

- Empregar `-Wall` e `-Werror`

☒ Empregar `-fsanitize=address`

3.11 GrupoJ

3.11.1 E4

- O tipo de dado pode ser int ou float, não tem como ser `TIPO_INDEFINIDO`
- 8 testes acusam problemas de alocação de memória

3.11.2 E3

3.11.3 E2

- Ao invés de usar `-Wno-unused-function`
  - Podem usar `%option noinput nounput` no scanner.l ao invés daquelas flags de compilação para contornar o aviso de compilação

3.11.4 E1

- Unificar a regra que tem `return yytext[0]`

☒ Colocar mais comentários.

- Todos os arquivos devem conter a identificação do grupo e os seus membros

☒ Comandos como `rm`, quando chamados no Makefile, podem vir acompanhados do parâmetro `-f` (force)

☒ Adicionar um alvo `clean` que remove arquivos gerados/compilados

- Empregar `-Wall` e `-Werror`

☒ Empregar `-fsanitize=address`

3.12 GrupoK

3.12.1 E4

- O tipo de dado está registrado em uma variável `int`, poderia ter sido um `enum`, com `typedef` para tornar o código mais legível.

3.12.2 E3

- Fatorar o código das ações no `parser.y` em situações onde é possível

3.12.3 E2

3.12.4 E1

☒ Sugiro que os {comment} e {blank} fiquem imediatamente antes da regra do ponto (return `TK_ER`). Mas acredito que o equívoco primordial é que a quebra de linha não está sendo ignorada. Podes submeter novamente com esta correção ASAP na mesma URL!

☒ Estes comandos precisaram ser executados para compilar sem warnings

```
sed -i 's/-lfl$//' Makefile
sed -i '6i%option noyywrap' scanner.l
sed -i '6i%option noinput nounput' scanner.l
sed -i 's#\";$#\"#' scanner.l
```

☒ Unificar a regra que tem `return yytext[0]`

☒ Comentar o código.

☒ Usar wildcards no Makefile (*it makes your life easier*)

- Todos os arquivos devem conter a identificação do grupo e os seus membros

☒ Comandos como `rm`, quando chamados no Makefile, podem vir acompanhados do parâmetro `-f` (force)

☒ Adicionar um alvo `clean` que remove arquivos gerados/compilados

☒ Empregar `-Wall` e `-Werror`

☒ Empregar `-fsanitize=address`

3.13 GrupoL

3.13.1 E4

- O tipo de dado está registrado em uma variável `int`, poderia ter sido um `enum`, com `typedef` para tornar o código mais legível.
- Sobre mensagens de erros
  - De uma maneira geral, faltou mencionar os nomes dos identificadores envolvidos em erros semânticos
- 2 testes acusam problemas de alocação de memória

3.13.2 E3

☒ Fatorar código no arquivo `scanner.l` pois existem trechos muito similares (para não dizer idênticos) em nas ações associadas aos tokens que geram valor.

- Fatorar o código das ações no `parser.y` em situações onde é possível

3.13.3 E2

☒ Acho sinceramente desnecessário juntar tudo no `all.o`

3.13.4 E1

☒ Comentar o código.

- Usar wildcards no Makefile (*it makes your life easier*)

☒ Todos os arquivos devem conter a identificação do grupo e os seus membros

☒ Comandos como `rm`, quando chamados no Makefile, podem vir acompanhados do parâmetro `-f` (force)

☒ Adicionar um alvo `clean` que remove arquivos gerados/compilados

☒ Empregar `-Wall` e `-Werror`

☒ Empregar `-fsanitize=address`

3.14 GrupoM

3.14.1 E4

- Flag -fsanitize=address não presente no log de compilação
- Todos os testes acusam problemas de alocação de memória

3.14.2 E3

- Fatorar o código das ações no `parser.y` em situações onde é possível

3.14.3 E2

3.14.4 E1

☒ Para que contar linhas se usaste `yylineno`?

  – Redundância de funcionalidades

☒ Usar wildcards no Makefile (*it makes your life easier*)

☒ Todos os arquivos devem conter a identificação do grupo e os seus membros

☒ Comandos como `rm`, quando chamados no Makefile, podem vir acompanhados do parâmetro `-f` (force)

☒ Adicionar um alvo `clean` que remove arquivos gerados/compilados

☒ Empregar `-Wall` e `-Werror`

☒ Empregar `-fsanitize=address` -

3.15 GrupoN

3.15.1 E4

- Não submetido.

3.15.2 E3

- Todos os testes falham pois ha um link gerado no parser.y linha 150, que por sua vez chama asd.c linha 13 e asd.c linha 15. Isso ocorre porque nesta linha o grupo decidiu criar nós na AST que estão fora da especificação da E3, tornando natural a presença do comentário "Não faço a menor ideia de como resolver esse memory leak AAAAAAAAAA". Esse nó nem deveria existir.
- Fazendo o fix temporário, isso causa uma série de problemas como em `declaracao_variavel_global`, principalmente porque não deveria haver o nó sendo pendurado na árvore mas também porque o `TK_ID` não foi liberado (ele não é usado). Talvez outros problemas possam estar presentes por razões semelhantes.
- Fatorar o código das ações no `parser.y` em situações onde é possível

3.15.3 E2

3.15.4 E1

☒ Todos os arquivos devem conter a identificação do grupo e os seus membros

☒ Comandos como `rm`, quando chamados no Makefile, podem vir acompanhados do parâmetro `-f` (force)

☒ Adicionar um alvo `clean` que remove arquivos gerados/compilados

☒ Empregar `-Wall` e `-Werror`

☒ Empregar `-fsanitize=address`

3.16 GrupoO

3.16.1 E4

- O tipo de dado está registrado em uma variável `int`, poderia ter sido um `enum`, com `typedef` para tornar o código mais legível.

3.16.2 E3

☒ Fatorar código no arquivo `scanner.l` pois existem trechos muito similares (para não dizer idênticos) em nas ações associadas aos tokens que geram valor.

- Fatorar o código das ações no `parser.y` em situações onde é possível

3.16.3 E2

- Ao invés de usar `-Wno-unused-function -Wno-unused-variable`
  - Podem usar `%option noinput nounput` no scanner.l ao invés daquelas flags de compilação para contornar o aviso de compilação
  - Remover as variáveis não utilizadas

3.16.4 E1

☒ Estes comandos precisaram ser executados para compilar sem warnings

```
sed -i 's/-lfl$//' Makefile
sed -i '6i%option noinput nounput' scanner.l
```

☒ Comentar o código.

☒ Usar wildcards no Makefile (*it makes your life easier*)

- Todos os arquivos devem conter a identificação do grupo e os seus membros

☒ Comandos como `rm`, quando chamados no Makefile, podem vir acompanhados do parâmetro `-f` (force)

☒ Adicionar um alvo `clean` que remove arquivos gerados/compilados

- Empregar `-Wall` e `-Werror`

☒ Empregar `-fsanitize=address`

3.17 GrupoP

3.17.1 E4

- Tipo de dado `S_NEVER` inexistente.

3.17.2 E3

- Fatorar o código das ações no `parser.y` em situações onde é possível

3.17.3 E2

☒ Usar o compilador `gcc` ao invés de `g++`

- OK, só usam para os testes.

3.17.4 E1

☒ Unificar a regra que tem `return yytext[0]`

☒ Comentar o código.

☒ Todos os arquivos devem conter a identificação do grupo e os seus membros

☒ Comandos como `rm`, quando chamados no Makefile, podem vir acompanhados do parâmetro `-f` (force)

☒ Adicionar um alvo `clean` que remove arquivos gerados/compilados

☒ Empregar `-Wall` e `-Werror`

☒ Empregar `-fsanitize=address`

3.18 GrupoQ

3.18.1 E4

- O tipo de dado está registrado em uma variável `int`, poderia ter sido um `enum`, com `typedef` para tornar o código mais legível.
- input e yyunput definidas mas não usadas, usar a opção do bison para evitar a definição

3.18.2 E3

☒ Relendo a especificação da E3 perceberão que 'inteiro', 'decimal', 'param', 'var', etc, coisas que envolvem tipos e declarações, não fazem parte da AST. Salvo claro, quando há inicialização de variável local, conforme ainda a especificação da E3. Esse equívoco faz com que todos os testes falhem. ;(

☒ Fatorar código no arquivo `scanner.l` pois existem trechos muito similares (para não dizer idênticos) em nas ações associadas aos tokens que geram valor.

☒ O `-fsanitize=address` não é empregado na hora de compilação com `make`

- Fatorar o código das ações no `parser.y` em situações onde é possível

3.18.3 E2

- É desnecessário fazer essa diferenciação entre *matched* e *unmatched* visto que não há ambiguidade no nosso else opcional.

3.18.4 E1

☒ #include serve unicamente para incluir cabeçalhos!

  - Então, remover a linha `#include "main.cpp"`

☒ Manter o arquivo main.c tal qual, não renomeá-lo para main.cpp

  - Não usar o g++, usar gcc
  - Nosso projeto é em C.

- Usar wildcards no Makefile (*it makes your life easier*)
- Todos os arquivos devem conter a identificação do grupo e os seus membros

☒ Comandos como `rm`, quando chamados no Makefile, podem vir acompanhados do parâmetro `-f` (force)

☒ Adicionar um alvo `clean` que remove arquivos gerados/compilados

- Empregar `-Wall` e `-Werror`
- Empregar `-fsanitize=address`

3.19 GrupoR

3.19.1 E4

- Não tem como ser do tipo dado `TIPO_INDEF`.

3.19.2 E3

☒ Fatorar código no arquivo `scanner.l` pois existem trechos muito similares (para não dizer idênticos) em nas ações associadas aos tokens que geram valor.

- Fatorar o código das ações no `parser.y` em situações onde é possível

3.19.3 E2

- Podem usar `%option noinput nounput` no scanner.l ao invés daquelas flags de compilação para contornar o aviso de compilação

3.19.4 E1

☒ Comentar o código.

☒ Todos os arquivos devem conter a identificação do grupo e os seus membros

☒ Comandos como `rm`, quando chamados no Makefile, podem vir acompanhados do parâmetro `-f` (force)

☒ Adicionar um alvo `clean` que remove arquivos gerados/compilados

☒ Empregar `-Wall` e `-Werror`

☒ Empregar `-fsanitize=address`

3.20 GrupoS

3.20.1 E4

- O tipo de dado está registrado em uma variável `int`, poderia ter sido um `enum`, com `typedef` para tornar o código mais legível.
- Flag -fsanitize=address não presente no log de compilação

3.20.2 E3

- Fatorar o código das ações no `parser.y` em situações onde é possível

3.20.3 E2

☒ Ainda sobrou um arquivo oculto com o nome `._main.c` com assinatura MAC

  – Removê-lo do versionamento e do `make tgz`

3.20.4 E1

☒ Remover todos os arquivos ocultos antes de submeter a solução

  – Arquivos ocultos são aqueles que tipicamente começam por ponto

☒ Remover os arquivos ocultos do mac

  – E não versioná-los no projeto

- Usar wildcards no Makefile (*it makes your life easier*)

☒ Todos os arquivos devem conter a identificação do grupo e os seus membros

☒ Comandos como `rm`, quando chamados no Makefile, podem vir acompanhados do parâmetro `-f` (force)

☒ Adicionar um alvo `clean` que remove arquivos gerados/compilados

☒ Empregar `-Wall` e `-Werror`

☒ Empregar `-fsanitize=address`

3.21 GrupoT

3.21.1 E4

- Grupo não submeteu o arquivo `errors.h` portanto o projeto não compilava. Professor providenciou a correção.
- Não tem como ser do tipo dado `UNDEFINED`

3.21.2 E3

- Fatorar código no arquivo `scanner.l` pois existem trechos muito similares (para não dizer idênticos) em nas ações associadas aos tokens que geram valor.
- Fatorar o código das ações no `parser.y` em situações onde é possível

3.21.3 E2

- Podem usar `%option noinput nounput` no scanner.l ao invés daquelas flags de compilação para contornar o aviso de compilação

3.21.4 E1

☒ Comentar o código.

- Usar wildcards no Makefile (*it makes your life easier*)

☒ Todos os arquivos devem conter a identificação do grupo e os seus membros

☒ Comandos como `rm`, quando chamados no Makefile, podem vir acompanhados do parâmetro `-f` (force)

☒ Adicionar um alvo `clean` que remove arquivos gerados/compilados

☒ Empregar `-Wall` e `-Werror`

☒ Empregar `-fsanitize=address`

3.22 GrupoV

3.22.1 E4

- Entrega em atraso.
- enum Type tem um tipo de dado que não deveria existir `TYPE_FUNCTION`
- 5 testes acusam problemas de alocação de memória

3.22.2 E3

- Fatorar código no arquivo `scanner.l` pois existem trechos muito similares (para não dizer idênticos) em nas ações associadas aos tokens que geram valor.
- Fatorar o código das ações no `parser.y` em situações onde é possível

3.22.3 E2

☒ *Talvez facilite sua correção eu te adiantar que chamamos o executavel de "analisador" no Makefile invés de "etapa2"*

3.22.4 E1

☒ Estes comandos precisaram ser executados para compilar sem warnings

```
sed -i '6i%option noinput nounput' scanner.l
```

☒ Comentar o código

☒ Usar wildcards no Makefile (*it makes your life easier*)

- Todos os arquivos devem conter a identificação do grupo e os seus membros

☒ Comandos como `rm`, quando chamados no Makefile, podem vir acompanhados do parâmetro `-f` (force)

☒ Adicionar um alvo `clean` que remove arquivos gerados/compilados

- Empregar `-Wall` e `-Werror`

☒ Empregar `-fsanitize=address`

3.23 GrupoW

3.23.1 E4

- O tipo de dado está registrado em uma variável `int`, poderia ter sido um `enum`, com `typedef` para tornar o código mais legível.

3.23.2 E3

- Fatorar código no arquivo `scanner.l` pois existem trechos muito similares (para não dizer idênticos) em nas ações associadas aos tokens que geram valor.
- Fatorar o código das ações no `parser.y` em situações onde é possível

3.23.3 E2

- Podem usar `%option noinput nounput` no scanner.l ao invés daquelas flags de compilação para contornar o aviso de compilação

3.23.4 E1

☒ Estes comandos precisaram ser executados para compilar sem warnings

```
sed -i '1d' Makefile
sed -i '1i\CC=gcc' Makefile
```

- Usar wildcards no Makefile (*it makes your life easier*)

☒ Evitar ños códigos fonte (removê-los antes de comitar)

- Todos os arquivos devem conter a identificação do grupo e os seus membros

☒ Comandos como `rm`, quando chamados no Makefile, podem vir acompanhados do parâmetro `-f` (force)

☒ Adicionar um alvo `clean` que remove arquivos gerados/compilados

- Empregar `-Wall` e `-Werror`

☒ Empregar `-fsanitize=address`

3.24 GrupoY

3.24.1 E4

Não submeteu.

3.24.2 E3

Não submeteu no prazo.

3.24.3 E2

Não submeteu.

3.24.4 E1

Não submeteu.

3.25 GrupoZ

3.25.1 E4

- Não tem como ser do tipo dado `TYPE_UNDEFINED`
- input e yyunput definidas mas não usadas, usar a opção do bison para evitar a definição

3.25.2 E3

☒ Fatorar código no arquivo `scanner.l` pois existem trechos muito similares (para não dizer idênticos) em nas ações associadas aos tokens que geram valor.

- Fatorar o código das ações no `parser.y` em situações onde é possível

3.25.3 E2

☒ Evitar de enviar binários compilados

  – Fazer `make clean` antes de `make tgz`

☒ Adicionar comentários para organizar as regras (parser.y)

3.25.4 E1

☒ Houve algum equívoco, vejam os testes.

- Todos os arquivos devem conter a identificação do grupo e os seus membros

☒ Comandos como `rm`, quando chamados no Makefile, podem vir acompanhados do parâmetro `-f` (force)

☒ Adicionar um alvo `clean` que remove arquivos gerados/compilados

- Empregar `-Wall` e `-Werror`

☒ Empregar `-fsanitize=address`

4 Estatísticas

4.1 Maior quantidade de erros esperados

O código zero na coluna "Erro" indica situação onde não há erro.

Erro	n	Porcento
ERR-WRONG-TYPE-ARGS	19	22.89
ERR-WRONG-TYPE	18	21.69
0	12	14.46
ERR-DECLARED	12	14.46
ERR-UNDECLARED	10	12.05
ERR-FUNCTION	4	4.82
ERR-VARIABLE	4	4.82
ERR-EXCESS-ARGS	2	2.41
ERR-MISSING-ARGS	2	2.41

4.2 Quais testes foram mais errados pelos grupos?

Test	Quantos.Grupos.Erraram	Erro.Esperado
qwe24	11	ERR-DECLARED
qwe69	9	ERR-WRONG-TYPE
qwe81	9	ERR-WRONG-TYPE
qwe82	9	ERR-WRONG-TYPE
qwe54	8	ERR-WRONG-TYPE-ARGS
qwe55	8	ERR-WRONG-TYPE-ARGS
qwe56	8	ERR-WRONG-TYPE-ARGS
qwe61	8	ERR-WRONG-TYPE-ARGS
qwe62	8	ERR-WRONG-TYPE-ARGS
qwe33	7	0
qwe49	7	ERR-WRONG-TYPE-ARGS
qwe50	7	ERR-WRONG-TYPE-ARGS
qwe51	7	ERR-WRONG-TYPE-ARGS
qwe52	7	ERR-WRONG-TYPE-ARGS
qwe53	7	ERR-WRONG-TYPE-ARGS
qwe57	7	ERR-WRONG-TYPE-ARGS
qwe58	7	ERR-WRONG-TYPE-ARGS
qwe59	7	ERR-WRONG-TYPE-ARGS
qwe60	7	ERR-WRONG-TYPE-ARGS
qwe63	7	ERR-WRONG-TYPE-ARGS
qwe64	7	ERR-WRONG-TYPE-ARGS
qwe06	6	0
qwe23	6	ERR-DECLARED
qwe41	6	ERR-FUNCTION
qwe78	6	ERR-WRONG-TYPE
qwe79	6	ERR-WRONG-TYPE
qwe80	6	ERR-WRONG-TYPE
qwe25	5	0
qwe26	5	0
qwe38	5	ERR-FUNCTION
qwe39	5	ERR-FUNCTION
qwe48	5	ERR-WRONG-TYPE-ARGS
qwe66	5	ERR-WRONG-TYPE
qwe12	4	ERR-UNDECLARED
qwe40	4	ERR-FUNCTION
qwe42	4	ERR-MISSING-ARGS
qwe43	4	ERR-MISSING-ARGS
qwe65	4	ERR-WRONG-TYPE
qwe67	4	ERR-WRONG-TYPE
qwe68	4	ERR-WRONG-TYPE
qwe10	3	ERR-UNDECLARED
qwe14	3	ERR-UNDECLARED
qwe15	3	ERR-UNDECLARED
qwe29	3	ERR-DECLARED
qwe31	3	0
qwe34	3	ERR-VARIABLE
qwe36	3	ERR-VARIABLE
qwe44	3	ERR-EXCESS-ARGS
qwe45	3	ERR-EXCESS-ARGS
qwe46	3	ERR-WRONG-TYPE-ARGS
qwe47	3	ERR-WRONG-TYPE-ARGS

5 Pesos

6 Final

7 Recuperação

Grupos em recuperação da E4:

- **Apenas grupos que entregaram no prazo** conforme regramentos
- Em Moodle já configurado para tal, use o link "Recuperação E4"

Grupo	E4.P
GrupoA	1
GrupoB	1
GrupoC	1
GrupoD	1
GrupoE	1
GrupoF	0.8
GrupoG	1
GrupoH	1
GrupoI	1
GrupoJ	1
GrupoK	0.8
GrupoL	1
GrupoM	1
GrupoO	1
GrupoP	1
GrupoQ	1
GrupoR	1
GrupoS	1
GrupoT	1
GrupoV	1
GrupoW	1
GrupoZ	1

Grupo	Etapa	E4.O	E4.S	E4.P
GrupoA	E4	9.04	7.62	1
GrupoB	E4	6.75	8.4	1
GrupoC	E4	9.64	9.18	1
GrupoD	E4	8.67	8.15	1
GrupoE	E4	9.52	8.7	1
GrupoF	E4	9.4	9.13	0.8
GrupoG	E4	1.45	5	1
GrupoH	E4	4.94	7.85	1
GrupoI	E4	9.52	8.7	1
GrupoJ	E4	6.39	8.6	1
GrupoK	E4	8.55	8.7	0.8
GrupoL	E4	8.8	8.42	1
GrupoM	E4	6.99	8	1
GrupoO	E4	9.64	9.45	1
GrupoP	E4	4.82	8.45	1
GrupoQ	E4	9.76	9.2	1
GrupoR	E4	9.76	9.45	1
GrupoS	E4	9.76	9.2	1
GrupoT	E4	9.88	9	1
GrupoV	E4	7.11	8.13	1
GrupoW	E4	9.52	8.95	1
GrupoZ	E4	9.64	9.2	1

- Política de recuperação ativada (vejam regras gerais)

$$\frac{x}{\text{GrupoG}}$$