COMP LabBook 2024 2 - E6

Lucas M. Schnorr

January 6, 2025

Contents

1 Introdução

1.1 Resultados corretos

Arquivo `e6_correct.csv`.

Espera-se que, para cada teste explicitado na coluna **Test**, o programa ASM gerado pelo compilador tenho o valor de retorno especificado na coluna **Value**, conforme o script bash disponibilizado na especificação da etapa.

Test	Value
kwj00	2
kwj01	13
kwj02	13
kwj03	13
kwj04	31
kwj05	16
kwj06	12
kwj07	108
kwj08	55
kwj09	46
kwj10	44
kwj11	32
kwj12	20
kwj13	2

Continued on next page

Continued from previous page

Test	Value
kwj14	11
kwj15	3
kwj16	3
kwj17	5
kwj18	5
kwj19	2
kwj20	4
kwj21	4
kwj22	3

1.2 Detalhes dos testes realizados

Arquivo `e6_detalhes.csv`

Cada linha representa um teste feito para o grupo identificado na coluna **Grupo** usando como entrada o arquivo **Test**. O valor da coluna **Alpha** é o valor de retorno visível pelo shell quando é executado o compilador do grupo (binário `etapa6`). O compilador do grupo gera um código em assembly que é compilado pelo `gcc`. O valor da coluna **Gamma** indica o valor de retorno visível pelo shell quando o `gcc` é executado, e normalmente indica se tudo foi bem (valor `0`) ou não (valor `1`). Uma vez o código assembly compilado para um binário executável, o valor da coluna **Beta** indica seu valor de retorno visível pelo shell. O valor em **Expected** é o valor de retorno esperado para aquele teste, conforme o código lá incluso.

Assume-se que se o assembly não for corretamente formado ou houver algum problema na compilação, o `gcc` informará através do valor `1`. Assim, a coluna **Correct** é TRUE (teste correto) se **Beta** for idêntico à **Expected**, **Alpha** for zero (compilação do compilador do grupo `etapa6` deu certo), e que **Gamma** é zero (compilação do código assembly funcionou sem problemas). Veja o arquivo `e6_detalhes.csv` completo no moodle.

2 Relatório de erros gerais

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

A primeira etapa de avaliação automática é a execução do compilador do grupo com uma data entrada. Estão listados os casos de falha de segmentação quando as entradas identificadas pela coluna `Test` da tabela abaixo são fornecidas para o compilador do grupo. Como o compilador é então incapaz de gerar ASM, estes testes são considerados falhos.

Grupo	Test
GrupoL	kwj00
GrupoL	kwj01
GrupoL	kwj02
GrupoL	kwj03
GrupoL	kwj04
GrupoL	kwj05
GrupoL	kwj06
GrupoL	kwj07
GrupoL	kwj08
GrupoL	kwj09
GrupoL	kwj10
GrupoL	kwj11
GrupoL	kwj12
GrupoL	kwj13
GrupoL	kwj14
GrupoL	kwj15
GrupoL	kwj16
GrupoL	kwj17
GrupoL	kwj18
GrupoL	kwj19
GrupoL	kwj20
GrupoL	kwj21
GrupoL	kwj22

2.2 Timeout da execução do compilador do grupo

Grupo	Test

2.3 Compilador GCC incapaz de reconhecer ASM gerado

Ao compilar o código ASM gerado pelo compilador do grupo, o compilador `gcc` relata algum tipo de problema nas instruções assembly geradas. Na tabela abaixo estão listados tais casos, para a entrada fornecida incialmente ao compilador do grupo identificada na coluna `Test`. Para identificar os problemas, pode-se executar:

```
etapa6 < test > test.s ; gcc test.s
```

Grupo	Test
GrupoL	kwj00
GrupoL	kwj01
GrupoL	kwj02
GrupoL	kwj03
GrupoL	kwj04
GrupoL	kwj05
GrupoL	kwj06
GrupoL	kwj07
GrupoL	kwj08
GrupoL	kwj09
GrupoL	kwj10
GrupoL	kwj11
GrupoL	kwj12
GrupoL	kwj13
GrupoL	kwj14
GrupoL	kwj15
GrupoL	kwj16
GrupoL	kwj17
GrupoL	kwj18
GrupoL	kwj19
GrupoL	kwj20
GrupoL	kwj21
GrupoL	kwj22
GrupoR	kwj05
GrupoR	kwj06
GrupoR	kwj07
GrupoR	kwj15
GrupoR	kwj17
GrupoR	kwj18
GrupoR	kwj19
GrupoR	kwj20
GrupoR	kwj22

2.4 Timeout da execução do programa ASM compilado

O programa ASM gerado pelo compilador do grupo `etapa6` e compilado para um binário executável com `gcc` não termina em um tempo aceitável de 3 segundos.

Grupo	Test

2.5 O código ASM gerado não faz o que o teste estimulava

Cada teste especificado na coluna `Test` tem um efeito, registrado nas variáveis locais ou globais. Este efeito pode ser oriundo de uma simples atribuição, de uma expressão aritmética, ou em função do controle de fluxo e chamadas de funções. Por exemplo, no teste `kwj27`, espera-se que o valor `12`, resultado da soma final, seja retornado pela função principal (observável através do shell, variável de ambiente `$?` consultada imeditamente após a execução do programa). Na tabela abaixo, nós temos os testes que foram considerados errados porque o código ASM gerado pelo compilador do grupo e compilado com `gcc` foi incapaz de gerar o valor especificado na coluna `Expected`, tendo gerado o valor observado na coluna `Beta`. São listados apenas 20 linhas do arquivo `e6_detalhes.csv`; consulte tal arquivo para a totalidade deste relatório de erros, ou veja os comentários abaixo onde os erros são apresentados por grupo.

Grupo	Test	Expected	Beta
GrupoF	kwj10	44	132
GrupoP	kwj07	108	139
GrupoL	kwj05	16	127

3 Histórico de comentários

3.1 DONE GrupoA

3.1.1 E6

- Sem comentários.

3.1.2 E5

- Sem comentários.

Could not parse R result

3.1.3 E4

- Evitar de quebrar a linha na mensagem de erro.

☐ Na regra `atribuicao`, faltou definir a inferência para o nó da AST correspondente (não é necessário invocar `tipagemDado`, apenas impor o tipo de quem recebe o valor para o nó da AST =).

☒ Melhorar a indentação do arquivo `parser.y`. Por exemplo, colocar comandos de C em sequência na linha, como é feito nas regras gramaticais de expressões, pode tornar difícil a leitura do código.

☒ As ações de várias regras possuem código repetido, isso poderia ser organizado em funções (Eng. Soft.) de maneira a tornar o programa mais legível.

3.1.4 E3

- Nenhum comentário.

3.1.5 E2

☒ Arquivo parser.y submetido na versão definitiva

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

3.1.6 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)

3.2 GrupoB

3.2.1 E6

- Não submetido.

3.2.2 E5

- Sem comentários.

Could not parse R result

3.2.3 E4

1. Normal

   - Agora compila.

   ☒ 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.

   ☒ Melhorar a indentação do arquivo `parser.y`. Por exemplo, colocar comandos de C em sequência na linha, como é feito em várias regras gramaticais, pode tornar difícil a leitura do código.

2. Preliminar

   ☒ Não compila pois falta `SIZE_MAX`

3.2.4 E3

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

3.2.5 E2

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

- Por exemplo, a listagem da estrutura do projeto

3.2.6 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.

3.3 DONE GrupoC

3.3.1 E6

- No caso o programa não lê da entrada padrão e não escreve na saída padrão, conforme especificado no enunciado. Assim, todos os testes falham. O professor analisou o código do grupo e verificou que se tornava demasiado complexo alterar em prol do grupo. Assim, o teste foi realizado de maneira diferente, adequando-se à não-adequação à especificação, de maneira a conduzir a avaliação.

3.3.2 E5

- Sem comentários.

Could not parse R result

3.3.3 E4

☒ Na regra de `atribuicao`, porque o grupo verifica se existem tipos incompatíveis se na realidade não existem compatíveis na especificação?

☒ Na regra `chamada_funcao`, a E4 explicita que nada deve ser feito para a chamada de função no que diz respeito aos tipos de dados. Por que o grupo então fez a definição do tipo do nó da chamada da função?

- O grupo conversou com o professor e explicou que fizeram isso como um bonus, um "a mais" na implementação pois sentiram falta da existência do tipo para a chamada de função pois esta faz parte de expressões.

☒ A implementação da "inferência" de tipos foi por copy-paste. Usar o conhecimento de Eng. Soft., encapsulando esse código em uma chamada de função, por exemplo.

☒ As ações de várias regras possuem código repetido, isso poderia ser organizado em funções (Eng. Soft.) de maneira a tornar o programa mais legível.

3.3.4 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).

- Grupo decidiu na E4 manter como não ponteiro

3.3.5 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`).

3.3.6 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

3.4 GrupoD

3.4.1 E6

- Não submetido.

3.4.2 E5

- Sem comentários

Could not parse R result

3.4.3 E4

☒ Melhorar a indentação do arquivo `parser.y`. Por exemplo, colocar comandos de C em sequência na linha, como é feito em várias regras gramaticais, pode tornar difícil a leitura do código.

☒ As ações de várias regras possuem código repetido, isso poderia ser organizado em funções (Eng. Soft.) de maneira a tornar o programa mais legível.

3.4.4 E3

1. Recuperação
   - ☐ 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).
     - Grupo decidiu por manter assim na E4.
   - ☐ Veja os pontos não marcados como feitos abaixo, na E1; entendo que eles ainda estão em aberto.
     - Grupo decidiu por manter sem modificar na E4
2. Normal
   - ☐ 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.

3.4.5 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.

3.4.6 E1

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

- E usar yytext[0] ao invés de explicitamente usar o literal
- Grupo decidiu por manter assim na E4

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

- Grupo decidiu por manter assim na E4

3.5 DONE GrupoE

3.5.1 E6

- Sem comentários.

3.5.2 E5

- Sem comentários

Could not parse R result

3.5.3 E4

☐ As ações de várias regras possuem código repetido, isso poderia ser organizado em funções (Eng. Soft.) de maneira a tornar o programa mais legível.

3.5.4 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).

3.5.5 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.

3.5.6 E1

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

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

3.6 DONE GrupoF

3.6.1 E6

- Ver testes falhos.

3.6.2 E5

- Sem comentários.

Could not parse R result

3.6.3 E4

☐ As mensagens de erro poderiam fazer referência aos lexemas dos símbolos envolvidos no erro, facilitando a identificação. Por exemplo, em kjl00, qual é o símbolo que não foi declarado?

☐ Na regra `atribuicao`, faltou definir a inferência para o nó da AST correspondente.

☒ Melhorar a indentação do arquivo `parser.y`. Por exemplo, colocar comandos de C em sequência na linha, como é feito nas regras gramaticais de expressões, pode tornar difícil a leitura do código.

☐ As ações de várias regras possuem código repetido, isso poderia ser organizado em funções (Eng. Soft.) de maneira a tornar o programa mais legível.

3.6.4 E3

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

3.6.5 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`.

3.6.6 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

3.7 DONE GrupoG

3.7.1 E6

- Sem comentários.

3.7.2 E5

- Sem comentários.

Could not parse R result

3.7.3 E4

☐ Na regra `chamada_funcao`, a E4 explicita que nada deve ser feito para a chamada de função no que diz respeito aos tipos de dados. Por que o grupo então fez a definição do tipo do nó da chamada da função?

☐ As ações de várias regras possuem código repetido, isso poderia ser organizado em funções (Eng. Soft.) de maneira a tornar o programa mais legível.

☒ Evitar uso de variáveis globais (tais como a `desired_kind`), ainda que funcione. Sempre há uma forma melhor.

3.7.4 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).

3.7.5 E2

- Nenhum comentário.

3.7.6 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

3.8 DONE GrupoH

3.8.1 E6

- Vejam os testes falhos.

3.8.2 E5

- Sem comentários.

Could not parse R result

3.8.3 E4

☐ Na regra `atribuicao`, faltou definir a inferência para o nó da AST correspondente.

☐ Na regra `chamadaFuncao`, a E4 explicita que nada deve ser feito para a chamada de função do que diz respeito aos tipos de dados. Por que o grupo então fez a definição do tipo do nó da chamada da função?

3.8.4 E3

☐ typedef é seu amigo para o tipo da árvore

  - Grupo decidiu na E4 ainda não usar typedef

3.8.5 E2

- Nenhum comentário.

3.8.6 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.

3.9 GrupoI

3.9.1 E6

- Não submetido.

3.9.2 E5

- Sem comentários.

Could not parse R result

3.9.3 E4

1. Em atraso
   - Submetido
   - Comentários feitos para a E3 não executados
2. No prazo
   - Não submetido.

3.9.4 E3

1. Em atraso

   ☐ 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).

   ☐ 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.

   ☐ 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).

2. Normal

   - Não submetido no prazo

3.9.5 E2

- Arquivos devem estar na raiz
  - O peso deste comentário aumentou

3.9.6 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.

3.10 DONE GrupoJ

3.10.1 E6

- Sem comentários.

3.10.2 E5

☐ problemas nas expressões aritméticas, algumas instruções usam temporários que nunca foram definidos.

Could not parse R result

3.10.3 E4

☐ Na regra `atribuicao_variavel`, faltou definir o tipo de dado para o nó da AST correspondente. Além disso, não é necessário validar pois aqui todos os tipos são compatíveis entre si, e, ainda mais, o tipo de dado da atribuição é imposta pelo tipo do identificador.

☐ As ações de várias regras possuem código repetido, isso poderia ser organizado em funções (Eng. Soft.) de maneira a tornar o programa mais legível.

3.10.4 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.

3.10.5 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.

3.10.6 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.

3.11 DONE GrupoK

3.11.1 E6

- Vejam os testes falhos.

3.11.2 E5

- Sem comentários.

Could not parse R result

3.11.3 E4

☒ Melhorar a indentação do arquivo `parser.y`. Por exemplo, colocar

comandos de C em sequência na linha, como é feito em todas as regras gramaticais de expressões, pode tornar difícil a leitura do código.

☒ As ações de várias regras possuem código repetido, isso poderia

ser organizado em funções (Eng. Soft.) de maneira a tornar o programa mais legível.

☐ Declarar as funções no escopo global, confirmar que elas estão

sendo corretamente verificadas (kjl04).

3.11.4 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).

3.11.5 E2

☒ Preliminar: arquivos devem estar na raiz

3.11.6 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.

3.12 DONE GrupoL

3.12.1 E6

- Programa compilador se termina com falha de segmentação.

3.12.2 E5

1. Atraso
   - Entregou atrasado.
2. Prazo
   - Não entregou.

3.12.3 E4

- Entregue com 2hs de atraso. Entregar no prazo.

☐ Na regra `comando_atribuicao`, faltou definir a inferência de tipo
para o nó da AST correspondente.

☐ Melhorar a indentação do arquivo `parser.y`. Por exemplo, colocar
comandos de C em sequência na linha, como é feito nas regras gramaticais de expressões, pode tornar difícil a leitura do código.

☐ As ações de várias regras possuem código repetido, isso poderia
ser organizado em funções (Eng. Soft.) de maneira a tornar o programa mais legível.

3.12.4 E3

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

3.12.5 E2

- Nenhum comentário.

3.12.6 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`).

3.13 DONE GrupoM

3.13.1 E6

- Sem comentários.

3.13.2 E5

☐ Usar a flag -I do compilador ao invés de incluir com caminhos
relativos tal como "`../include/iloc.h`".

Could not parse R result

3.13.3 E4

☐ Na regra `function_call`, a E4 explicita que nada deve ser feito
para a chamada de função no que diz respeito aos tipos de dados. Por que o grupo então fez a definição do tipo do nó da chamada da função?

☐ As ações de várias regras possuem código repetido, isso poderia
ser organizado em funções (Eng. Soft.) de maneira a tornar o programa mais legível.

3.13.4 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).

- Grupo decidiu na E4 manter tal qual.

☒ 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.

3.13.5 E2

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

☒ Usar wildcards do makefile para simplificar as regras

3.13.6 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

3.14 GrupoN

3.14.1 E6

- Não submetido.

3.14.2 E5

- Não entregou.

3.14.3 E4

1. Recuperação
   - Não submetido
2. Normal

   ☐ Falha de segmentação em vários testes

   ☐ Nenhum erro semântico detectado

   ☐ 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. (segundo aviso)

   ☐ O professor identificou que muito pouco código é referente à E4,

   quando comparando, por exemplo, contra a E3 do grupo.

   ☐ Foi criado um módulo chamado `comp_utils` que agora contém código da

   parte da AST (E3). A sugestão é manter um módulo bem caracterizado pois algo "utils" é genérico sem funcionalidade específica. Essas mudanças dificultam também a avaliação pelo professor. :-(

   ☐ Na regra `func`, atenção porque o identificador da função está

   sendo incluído na tabela local e não na global, onde deveria ser. Antes de `pushLocalTable`, deve haver inserção do `TK_IDENTIFICADOR` na tabela global. Sugiro reverificar os pontos de empilhamento e desempilhamento.

   ☐ Não foi encontrada instrumentação da AST para inferência de

   tipos (o nó da AST não possui campo de tipo de dado).

   ☐ A estrutura `ast_token` também tem campos não solicitados pela

   especificação. Talvez pudessem fazer uma limpeza de coisas desnecessárias.

3.14.4 E3

1. Recuperação

   ☐ 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, como para operadores simples e compostos.

   ☐ Veja os pontos não marcados como feitos abaixo, na E2; entendo

   que eles ainda estão em aberto.

2. Normal

☐ 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.

3.14.5 E2

☐ Melhorar o makefile para que se possa usufruir de um sistema de

compilação que permita compilação parcial dos vários fontes do projeto.

3.14.6 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

3.15 DONE GrupoO

3.15.1 E6

- Vejam os testes falhos.

3.15.2 E5

- Sem comentários.

Could not parse R result

3.15.3 E4

☐ Na inferência de tipos das expressões, emprega-se a `typeInfer`,

que recebe apenas um nó da AST e faz a inferência baseado nos filhos. O problema é que naquelas regras, essa função é chamada apenas com o terceiro filho como parâmetro. Ou seja, a inferência é feita somente naquele filho, e não no nó que está sendo criado. A inferência do nó que está sendo criada é portanto postergada. Qual a razão de se implementar assim?

☐ As ações de várias regras possuem código repetido, isso poderia

ser organizado em funções (Eng. Soft.) de maneira a tornar o programa mais legível.

3.15.4 E3

☒ Veja os pontos não marcados como feitos abaixo, na E1; entendo

que eles ainda estão em aberto.

3.15.5 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.

3.15.6 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

3.16 DONE GrupoP

3.16.1 E6

- Os códigos gerados pelo compilador são transformados em executáveis pelo gcc, mas a execução gera falha de segmentação (em todos os testes).

3.16.2 E5

- Sem comentários.

Could not parse R result

3.16.3 E4

☒ Poderia implementar uma função para a inferência, evitando cópia

de código.

☒ As ações de várias regras possuem código repetido, isso poderia

ser organizado em funções (Eng. Soft.) de maneira a tornar o programa mais legível.

3.16.4 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.

3.16.5 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.

3.16.6 E1

☐ Melhorar o makefile para que se possa usufruir de um sistema de

compilação que permita compilação parcial dos vários fontes do projeto.

- Nada novo aqui na E4

3.17 GrupoQ

3.17.1 E6

- Não submetido.

3.17.2 E5

- Não submetido.

3.17.3 E4

- Não submetido

3.17.4 E3

- Não submetido

3.17.5 E2

- Não submetido

3.17.6 E1

- Não submetido

3.18 DONE GrupoR

3.18.1 E6

- Ver os testes falhos.

3.18.2 E5

- Sem comentários.

Could not parse R result

3.18.3 E4

☐ As ações de várias regras possuem código repetido, isso poderia

ser organizado em funções (Eng. Soft.) de maneira a tornar o programa mais legível.

3.18.4 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).

3.18.5 E2

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

um número de linha

3.18.6 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`

3.19 DONE GrupoS

3.19.1 E6

- Problema na implementação do & lógico?

3.19.2 E5

- Sem comentários.

Could not parse R result

3.19.3 E4

- Boa organização do código em módulos

☒ Minha sugestão seria para corrigir a indentação, padronizando-a,

sobretudo no arquivo parser.y. Por exemplo, na regra `atribuicao`, temos um problema de indentação (em outras regras com comandos if com else if, e if com else também).

☒ As ações de várias regras possuem código repetido, isso poderia

ser organizado em funções (Eng. Soft.) de maneira a tornar o programa mais legível.

3.19.4 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).

3.19.5 E2

☒ Nos comentários que categorizam as regras gramaticais, usar texto

não em uppercase total.

3.19.6 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.

3.20 GrupoT

3.20.1 E6

- Não submetido.

3.20.2 E5

- Sem comentários.

Could not parse R result

3.20.3 E4

☒ Usar a flag -I do compilador ao invés de incluir com caminhos

relativos tal como "`../include/data_structures.h`".

☒ Melhorar a indentação do arquivo `parser.y`. Por exemplo, colocar

comandos de C em sequência na linha, como é feito nas regras gramaticais de expressões, pode tornar difícil a leitura do código.

☒ As ações de várias regras possuem código repetido, isso poderia

ser organizado em funções (Eng. Soft.) de maneira a tornar o programa mais legível.

3.20.4 E3

☒ Veja os pontos não marcados como feitos abaixo, na E2; entendo

que eles ainda estão em aberto.

3.20.5 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.

3.20.6 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

3.21 GrupoZ

3.21.1 E6

- Não submetido.

3.21.2 E5

- Sem comentários.

Could not parse R result

3.21.3 E4

- Não submetido.

3.21.4 E3

1. Submissão em atraso

   ☐ 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).

3.21.5 E2

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

- Realmente ficou faltando.

3.21.6 E1

☒ Melhorar o makefile para que se possa usufruir de um sistema de

compilação que permita compilação parcial dos vários fontes do projeto.

4 Pesos

Grupo	E6.P
GrupoA	1
GrupoC	1
GrupoE	1
GrupoF	1
GrupoG	1
GrupoH	1
GrupoJ	1
GrupoK	1
GrupoL	1
GrupoM	1
GrupoO	1
GrupoP	1
GrupoR	1
GrupoS	1

5 Final

Grupo	Etapa	E6.O	E6.S	E6.P
GrupoA	E6	10	9	1
GrupoE	E6	10	9	1
GrupoG	E6	10	9	1
GrupoJ	E6	10	9	1
GrupoM	E6	10	9	1
GrupoS	E6	9.6	8.8	1
GrupoH	E6	8.7	8.4	1
GrupoO	E6	8.3	8.4	1
GrupoC	E6	8.3	7.8	1
GrupoK	E6	7.4	8	1
GrupoR	E6	5.7	7.4	1
GrupoF	E6	1.3	5.8	1
GrupoP	E6	0	6.2	1
GrupoL	E6	0	5	1

6 Recuperação

Não haverá conforme combinado.