2022. DESENVOLVIMENTO GUIADO POR TIPOS

MCTA016-13
Permalink

Turmas: DA1MCTA016-13SA e NA1MCTA016-13SA
Professor: Emilio Francesquini
E-mail: e.francesquini@ufabc.edu.br

1 Avisos

- [2022-06-02 Thu] Página online

2 Apresentação

Em sistemas de computação com aplicações sensíveis (área financeira, lançamento de um foguete, etc...) busca-se por códigos comprovadamente livres de falhas. O modelo tradicional para assegurar a qualidade do software se baseia na detecção de falhas durante a execução através da criação e execução de testes automatizados. Este método, por sua natureza, garante apenas que especificamente o que foi testado funciona conforme verificado pelos testes. Como o número de testes é finito, tais técnicas não raramente são combinadas com a avaliação da cobertura de código. Ainda assim, este método não é capaz de provar que não há falhas assim como não impede que códigos incorretos sejam escritos. Ademais, para qualquer software de tamanho razoável a complexidade ciclomática para que todos os caminhos de execução sejam verificados torna a escrita de testes de maneira manual proibitivamente complexa.

Um modelo complementar é o da utilização de tipos e funções com restrições de tal forma a maximizar a localização de erros em tempo de compilação, evitando que eles ocorram durante a execução. Esta abordagem evita a escrita de diversos testes automatizados que, por definição, perdem o seu sentido em uma linguagem com tipagem mais restritiva já que o compilador ou prova que tais situações são impossíveis ou emite um erro de compilação.

3 Objetivos

Neste curso sobre Programação Guiada por Tipos nós vamos explorar em detalhes as principais técnicas que são empregadas para evitar erros durante a execução. Para isto utilizaremos ferramentas que, em sua maioria, ainda não estão presentes nas principais linguagens utilizadas hoje no mercado.

Ao fim do curso, você deverá ser capaz de criar aplicações que, se não forem 100% *type safe*, pelo menos serão capazes de evitar vários erros em tempo de execução.

Se conseguimos salvar pelo menos um de vocês de terem que trabalhar na véspera de natal para atender um chamado por um pau que deu em produção, já vai ter valido a pena! :-D

---

4 Pré-requisitos

É exigido que os participantes tenham uma ótima familiaridade com programação. A linguagem de programação utilizada é Haskell, logo familiaridade com programação funcional e Haskell é um diferencial!

Se Haskell ou programação funcional não é o seu forte, aproveite para fazer uma maratona dos nossos vídeos sobre programação funcional em Haskell e seguir o material didático que acompanham esses vídeos disponível aqui.

---

5 Sobre a Disciplina

MCZA053-22 Desenvolvimento Guiado por Tipos

- TPI 4-0-4
- Recomendação: Processamento da Informação; Paradigmas de Programação; Programação

Estruturada

> **Objetivos** Permitir a identificação de padrões comuns durante a fase de planejamento e desenvolvimento de sistemas e a manipulação de Tipos de Dados Algébricos e Tipos Dependentes de forma a permitir a criação de códigos concisos, genéricos e, em sua maioria, comprovadamente corretos.
>
> **Ementa** Revisão de Programação Funcional, Functores - Monoides, Mônadas, Comônadas, Estruturas de Dados Funcionais, Tipos de dados algébricos generalizados, Tipos Dependentes.

Fonte: Catálogo de Disciplinas. Pró-reitoria de Graduação, UFABC 2021-2022 V.2

---

6 Informações Gerais

- **Oferecimentos**: DA1MCZA053-22SA
- **Assistente de ensino:** Matheus Fernandes `@Matheus Fernandes#6268` no Discord.
    - Disponível para tirar dúvidas da disciplina e ajudar (conversando, dando ideias e dicas, mas não pondo a mão na massa!) no desenvolvimento das listas e do projeto.

Esse oferecimento da disciplina será completamente online, incluindo aulas, plantões e avaliações.

> **Note**
>
> Todas as aulas, com e sem participação dos alunos, serão gravadas e disponibilizadas online segundo a Licença Creative Commons Atribuição-NãoComercial 4.0 Internacional (CC-BY-NC). Todos os participantes do curso dão sua tácita e irrevogável autorização para que suas imagens e falas sejam transmitidas, gravadas e editadas segundo a licença acima pelo docente responsável, sem nenhuma cobrança, para uso em distintos canais de comunicação e peças publicitárias sem fins comerciais.

6.1 Dinâmica de ensino

Serão disponibilizadas videoaulas com explicação do conteúdo teórico além do conteúdo prático com codificação em tempo real. Como material de apoio os alunos terão acesso aos códigos desenvolvidos durante as aulas, além de poderem contar com o uso do Discord (veja abaixo) para tirar dúvidas e discutir assuntos pertinentes à disciplina.

Semanalmente serão disponibilizadas listas de exercícios sobre o conteúdo apresentado. As entregas destas listas serão consideradas como uma avaliação e deverão ser feitas em até uma semana após a sua disponibilização. As notas dessas atividades serão utilizadas para a composição da média final. As aulas não serão dadas em tempo real devido aos inúmeros possíveis problemas técnicos, já que são mais de 75 alunos de alunos matriculados na turma atualmente. Nos dias e horários em que haveria aula e atendimento, estarei online na ferramenta Discord (que além de chat, faz captura de voz e tela) para tirar dúvidas sobre o conteúdo previsto para aquela data.

---

7 Dias, horários e local das aulas

> **Tip**
>
> Inscreva-se o quanto antes no servidor do Discord da disciplina (https://discord.gg/JSgnfdE). Todos os anúncios e comunicações serão feitos por lá.

Figure 1: Servidor do Discord da disciplina.

Os professor **estará online no Discord** para tirar dúvidas nos seguintes horários:

- **Segundas**
  - 11h00 às 12h00
  - 19h00 às 20h00
- **Quintas**
  - 09h00 às 10h00
  - 21h00 às 22h00

Eventuais dúvidas e questionamentos poderão ser enviados em outros horários. Contudo, fora dos horários acima, o professor pode não atendê-los prontamente devido às suas outras atividades.

Os vídeos das aulas serão disponibilizados online, na seguinte página:
http://haskell.pesquisa.ufabc.edu.br/

Eventualmente também teremos aulas síncronas, com live coding. Essas aulas serão anunciadas tanto nesta página quanto no Discord (link acima).

---

8 Datas importantes

8.1 Listas semanais

Teremos listas semanais que serão utilizadas para a avaliação. Cada lista semanal é composta de um número variado (não muito grande) de exercícios de programação:

- Os exercícios semanais de programação serão divididos em: 50% nível fácil, 25% nível intermediário e 25% nível desafiador

As datas de liberação dos enunciados e prazo para entrega são listados abaixo:

Lista	Enunciado/Prazo	Resolução	Link (Matriculados e ouvintes)

Lista	Enunciado/Prazo	Resolução	Link (Matriculados e ouvintes)
1	10/06 - 20/06		
2	17/06 - 27/06		
3	24/06 - 04/07		
4	01/07 - 11/07		
5	08/07 - 18/07		
6	15/07 - 25/07		
7	22/07 - 01/08		
8	29/07 - 08/08		
9	05/08 - 15/08		
10	12/08 - 22/08		

Warning

• Todas as entregas devem ser feitas **exclusivamente pelo Github** Classroom através dos links acima. Listas enviadas por outros meios não serão aceitas.
• Entrega toda segunda-feira até às 23:59. Listas entregues com atraso de até um dia serão aceitas, contudo com um desconto de 50% da nota.
• O código submetido **precisa** compilar ao executar `stack test`
• Caso não consiga fazer algum exercício coloque a palavra-chave `undefined` como definição da função. Ex.: `isEven = undefined`
• A execução de `stack test` não pode travar e nem entrar em loop infinito.

A falha em qualquer um desses casos acarretará em nota 0 na lista inteira. Assistam os vídeos com as correções das listas de exercício (em especial das listas 1 e 2) dos oferecimentos anteriores da disciplina que podem ser encontrados nos links listados acima.

8.2 Projeto

Data para entrega: **28/08/2022 23h59**

- Entregas via Github Classroom. O link estará aqui em breve.
- O projeto é **individual**[1].
- Todos os *commits* e o *push* final devem ter sido feitos até no máximo às **23h59 do dia 28/08/2022**.
    - Atenção, caso ocorram/commits/pushes após o prazo, será considerada apenas a última versão para avaliação. Caso o último esteja com atraso > 3 dias, então será considerado o

último commit até 3 dias (valendo no máximo 5).
    - Entregas em atraso sofrerão descontos conforme a tabela abaixo.

Dias em atraso	Nota máxima
1 dia	7
2 dias	6
3 dias	5
>3 dias	0

- O conteúdo do projeto é bem dinâmico e aberto podendo ser:
    - um estudo bibliográfico
    - reprodução de um artigo científico (consulte o professor sobre a validade do artigo escolhido antes de começar)
    - implementação de um projeto completo (que pode ser o que o aluno desejar) usando boa parte (senão todas) as técnicas apresentadas durante o curso
    - comparações entre linguagens entre maneiras que type-safety podem ser alcançadas
    - ...

- No mesmo repositório do seu código, deverá haver um relatório de no máximo 1 páginas (em PDF! nada de `.md`, `.docx`, `.odt`, ou qualquer outro formato da moda...) descrevendo:
    - Qual é o seu projeto
    - Como utilizar (se for qualquer coisa diferente de `stack run`) o seu código (se for um código)
    - Dificuldades, surpresas e destaques do seu projeto
- O seu relatório também deverá conter **obrigatoriamente** um link para um vídeo que:
    - Esteja abrigado no Youtube (ou qualquer serviço semelhante)
    - Seja privado/não listado e disponível apenas àqueles com o link
    - Tenha no máximo 3 minutos (ou seja, <= 180 segundos). **Sem exceções**.
    - Faça uma brevíssima explicação do que se trata o seu projeto
- Durante a correção o professor pode pedir por esclarecimentos adicionais.

Que tipos de projetos serão aceitos?

Lembre-se que você terá o quadrimestre todo para entregar o seu projeto. Logo é esperado que o seu projeto reflita um esforço de desenvolvimento compatível com o prazo disponível.

Caso esteja com dúvidas se a sua ideia é o suficiente para um projeto final que será bem avaliado converse com o professor o quanto antes!

---

9 Aulas

Neste oferecimento, a disciplina será online e se utilizará dos vídeos criados pelos professores Emilio Francesquini e Fabrício Olivetti. Por isto todo o material das aulas (slides, vídeos, textos) serão

compartilhados. O site com esse material compartilhado está disponível em:

https://haskell.pesquisa.ufabc.edu.br/desenvolvimento-orientado-a-tipos/

A playlist com todos os vídeos das aulas teóricas do curso pode ser encontrada aqui:

https://www.youtube.com/watch?v=KTmHPTxKIBY&list=PLYItvall0TqKaY6qObQMlZ45Bo94xq9Ym

Assine, clique o sininho coisa e tal para ser notificado!

Para facilitar, na tabela abaixo você encontra os assuntos juntamente com links diretos para o conteúdo e os vídeos além de uma sugestão sobre a semana ideal para cada tópico.

Semana Sugerida	Tópico	Material de Estudo
1	Breve revisão de Haskell, tipos, restrições (*constraints*)	https://haskell.pesquisa.ufabc.edu.br/desenvolvimento-orientado-a-tipos/01.introducao/
2	Polimorfismo paramétrico e *ad-hoc*	https://haskell.pesquisa.ufabc.edu.br/desenvolvimento-orientado-a-tipos/02.tipos/
3	Tipos de dados algébricos (*algebraic data types* - ADTs)	https://haskell.pesquisa.ufabc.edu.br/desenvolvimento-orientado-a-tipos/03.adts/
3	Tipos fantasma	https://haskell.pesquisa.ufabc.edu.br/desenvolvimento-orientado-a-tipos/03.adts/#tipo-fantasma
4	Tipos, *Kinds*, *Sorts*	https://haskell.pesquisa.ufabc.edu.br/desenvolvimento-orientado-a-tipos/04.kinds/
5	Tipos de dados algébricos generalizados (*Generalized Algebraic Data Types - GADTs*)	https://haskell.pesquisa.ufabc.edu.br/desenvolvimento-orientado-a-tipos/05.gadts/
6	Estudo de caso: vetores indexáveis seguros	https://haskell.pesquisa.ufabc.edu.br/desenvolvimento-orientado-a-tipos/05.gadts/#estudo-de-caso-vetores-index%C3%A1veis-seguros
6	Singleton types	https://haskell.pesquisa.ufabc.edu.br/desenvolvimento-orientado-a-tipos/05.gadts/#um-cheiro-de-singleton-types
6	Tipos Existenciais	https://haskell.pesquisa.ufabc.edu.br/desenvolvimento-orientado-a-tipos/05.gadts/#tipos-existenciais
7	Type Families, Closed Type Family	https://haskell.pesquisa.ufabc.edu.br/desenvolvimento-orientado-a-tipos/06.typefamily/
7	Criando uma lista heterogênea	https://haskell.pesquisa.ufabc.edu.br/desenvolvimento-orientado-a-tipos/06.typefamily/#lista-heterog%C3%AAnea
8	Open Type Family	https://haskell.pesquisa.ufabc.edu.br/desenvolvimento-orientado-a-tipos/06.typefamily/#open-type-family

Semana Sugerida	Tópico	Material de Estudo
8	Tipos associados	https://haskell.pesquisa.ufabc.edu.br/desenvolvimento-orientado-a-tipos/06.typefamily/#tipos-associados
9	Data family	https://haskell.pesquisa.ufabc.edu.br/desenvolvimento-orientado-a-tipos/06.typefamily/#data-family
9	Estudo de caso: validação de strings	https://haskell.pesquisa.ufabc.edu.br/desenvolvimento-orientado-a-tipos/06.typefamily/#estudo-de-caso-valida%C3%A7%C3%A3o-de-strings
10	Tipos dependentes	-
11	Discussão e implementação de projetos	-
12	Discussão e implementação de projetos	-

Dúvidas, pelo Discord: https://discord.gg/DtQtBWvwzg

10 Critério de avaliação

Honestidade Acadêmica

Entre outros, o código de ética da UFABC estabelece em seu artigo 25 que é **eticamente inaceitável** que os discentes:
I - fraudem avaliações;
II - fabriquem ou falsifiquem dados;
III - plagiem ou não creditem devidamente autoria;
IV - aceitem autoria de material acadêmico sem participação na produção;
V - vendam ou cedam autoria de material acadêmico próprio a pessoas que não participaram da produção.

Muitos ainda têm dúvidas sobre a interpretação das regras definidas pelo Código de Ética da UFABC. Por esta razão, diversos professores elaboraram um documento (**disponível aqui**) com vários exemplos e esclarecendo a interpretação das regras acima. Abaixo uma versão resumida. **Sempre consulte o documento completo ou converse com o seu professor em caso de dúvidas!**

> - **Regra 1** - Você não pode enviar para avaliação um trabalho que não seja de sua própria autoria ou que seja derivado/baseado em soluções elaboradas por outros.
> - **Regra 2** - Você não pode compartilhar a sua solução com outros alunos nem pedir aos seus colegas que compartilhem as soluções deles com você.
> - **Regra 3** - Nos trabalhos enviados para avaliação você deve indicar eventuais assistências que você tenha recebido.
>
>   ATENÇÃO: todos os trabalhos enviados para avaliação poderão ser verificados por um sistema automatizado de detecção de plágio.
>
> Qualquer violação às regras descritas acima implicará:
>
> - Descarte dos conceitos atribuídos a TODAS as tarefas avaliativas regulares de TODOS os envolvidos, causando assim suas reprovações automáticas com conceito F .
> - Possível denúncia à Comissão de Transgressões Disciplinares Discentes da Graduação, a qual decidirá sobre a punição adequada à violação que pode resultar em advertência, suspensão ou desligamento , de acordo com os artigos 78-82 do Regimento Geral da UFABC.
> - Possível denúncia apresentada à Comissão de Ética da UFABC, de acordo com o artigo 25 do Código de Ética da UFABC.

A avaliação da disciplina será composta pelas seguintes notas:

- N_P é a nota do projeto
- N_L é a nota das listas de exercícios semanais

A nota N_L será calculada como a média aritmética das listas semanais, cada lista valendo 10 pontos, descartando-se a lista com a pior nota. Ou seja, se por qualquer motivo você não puder enviar uma das listas ou caso tenha sido avaliada como uma nota baixa, basta que faça todas as demais para que não haja prejuízo na sua nota.

A nota final (N_F) será determinada pela **média harmônica ponderada ** de N_P e N_L, com pesos 6 e 4 respectivamente:

$$N_F = \frac{10}{\frac{6}{\max\{0.1, N_P\}} + \frac{4}{\max\{0.1, N_L\}}}$$

O conceito final (C_F) será obtido de acordo com a equação abaixo:

$$C_F = \begin{cases} \mathbf{F}, & \text{se } N_F \in [0,0; 5,0) \\ \mathbf{D}, & \text{se } N_F \in [5,0; 6,0) \\ \mathbf{C}, & \text{se } N_F \in [6,0; 7,0) \\ \mathbf{B}, & \text{se } N_F \in [7,0; 8,5) \\ \mathbf{A}, & \text{se } N_F \in [8,5; 10,0] \end{cases}$$

Caso seja verificado ocorrência de plágio no projeto final, o aluno será automaticamente reprovado com F

10.1 Recuperação

A resolução ConsEPE nº 182 assegura a todos os alunos de graduação com C_F igual a **D** ou **F** o direito a fazer uso de mecanismos de recuperação.

A recuperação será feita através de uma nova entrega do projeto final levando em consideração a correção do projeto original e seguido de uma entrevista com o docente a ser marcada entre os dias 08/09/2022 e 09/09/2022. A sua nota será utilizada para compor a o conceito pós-recuperação C_R conforme as equações abaixo:

$$N_R = \frac{P_R + N_F}{2}$$

Caso 1 $C_F = D$**:**

$$C_R = \begin{cases} \mathbf{C}, & \text{se } N_R \geq 6,0 \\ \mathbf{D}, & \text{caso contrário} \end{cases}$$

Caso 2 $C_F = F$**:**

$$C_R = \begin{cases} \mathbf{D}, & \text{se } N_R \geq 5,0 \\ \mathbf{F}, & \text{caso contrário} \end{cases}$$

10.2 Regulamentações Relevantes

- Resolução ConsEPE Nº 240 - Estabelece a autorização para a oferta excepcional de componentes curriculares e de outras atividades acadêmicas remotas durante o(s) chamado(s) “Quadrimestre(s) Suplementar(es)”
- Resolução ConsEPE nº 182 - Regulamenta a aplicação de mecanismos de recuperação
- Código de Ética da UFABC
- Resolução ConsUni nº 63 - Regimento Geral da UFABC

---

11 Recursos Online

11.1 Grupos, listas, páginas, ...

- Grupo de Estudos em Haskell da UFABC
  - **Não esqueça de participar do Discord**: https://discord.gg/JSgnfdE
- Haskell Home Page
- #haskell IRC channel
- StackOverflow
- A lista Haskell-beginners é um bom lugar para buscar respostas para perguntas básicas
- A lista Haskell-cafe é boa para buscar respostas a perguntas mais elaboradas
- O Haskell Wiki tem uma lista de respostas às perguntas mais comuns

11.2 Disciplinas em Haskell

- MCTA016-13: Paradigmas de Programação (em Haskell), UFABC. 2021, 2020, 2019, 2018.
- CR062-Programação Funcional em Haskell, UFABC. 2019, 2018.
- Estruturas de Dados Puramente Funcionais. 2019.
- G51PGP: Programming Paradigms, University of Nottingham. 2019.
- CS653: Functional Programming, Indian Institute of Technology Kanpur. 2018.
- CIS 194: Introduction to Haskell, University of Pennsylvania. 2016, 2015, 2014, 2013.
- ... mais exemplos aqui

11.3 Leituras

- Haskell Wikibook
- A Gentle Introduction to Haskell
- Haskell Cheat Sheet
- What I Wish I Knew When Learning Haskell

11.4 Documentação

- Standard library documentation
- Hackage - Repositório de pacotes
- Hoogle - Procurando funções para as quais você não sabe o nome?
- Hayoo - Parecido com o Hoogle, porém mais completo (procura todo o Hackage, mas está frequentemente fora do ar...)
- Se você realmente precisar, a linguagem e a biblioteca padrão do Haskell são definidos pelo Haskell 2010 - Language Report

---

12 Bibliografia

Este curso é um apanhado de uma série de livros. Caso queira seguir um livro específico recomendo:

- MAGUIRE, Sandy. Thinking with Types. Lulu, 2018.

12.1 Bibliografia Básica

- BRADY, Edwin. Type-driven development with Idris. Manning Publications Company, 2017.
- OKASAKI, Chris. Purely functional data structures. Cambridge University Press, 1999.
- SPIVAK, David I. Category theory for the sciences. MIT Press, 2014.

12.2 Bibliografia Complementar

- FRIEDMAN, Daniel P.; CHRISTIANSEN, David Thrane. The Little Typer. MIT Press, 2018.
- GRAHAM, H. Programming in Haskell. 2007.
- MILEWSKI, Bartosz. Category theory for programmers. Blurb, 2018. (PDF gratuito)
- SITNIKOVSKI, Booro. Gentle Introduction to Dependent Types with Idris. Scotts Valley: CreateSpace Independent Publishing, 2018.

13 Responsáveis, contatos, etc...

O curso foi desenvolvido pelos Profs. Fabrício Olivetti e Emilio Francesquini da Universidade Federal do ABC.

Contatos:

- Discord (método preferencial!): https://discord.gg/DtQtBWvwzg
- Emilio Francesquini - e.francesquini@ufabc.edu.br

---

1. Pode fazer em grupo? Sim. Grupos de no máximo uma pessoa! 😝 ↩