Relatório

Luis Felipe F. Cardoso1, Luiz Antonio L. de F. Leite1, Luiz Sérgio S. Maciel Filho1, Max Jose L. Pantoja Jr.1, Wesley P. Barbosa1

1Instituto de Ciências Exatas e Naturais – Universidade Federal do Pará (UFPA) – Belém – PA – Brazil

@icen.ufpa.br, @icen.ufpa.br, luiz.filho@icen.ufpa.br, @icen.ufpa.br, @icen.ufpa.br

**Abstract.** This report describes the implementation of regular expressions using the Python programming language to validate different kinds of inputs on a form, such as names, email addresses, telephone numbers, datetime values, cpf numbers, passwords, and numbers, as well as different kinds of family configurations given a set of gender and age rules as stablished on the instructions for the regular expressions group assignment of the “Linguagens Formais, Autômatos e Computabilidade” course for the 2023.2 class of the Computer Science undergraduate program at Universidade Federal do Pará.

**Resumo.** Este relatório descreve a implementação de expressões regulares na linguagem de programação Python para reconhecer diferentes tipos de entradas em um formulário, como nomes, endereços de email, números de telefone, valores de datetime, números de cpf, senhas e números, bem como diferentes tipos de arranjos familiares levando em consideração determinadas regras relativas a sexo biológico e idade, como estipulado nas instruções para a atividade em grupo de expressões regulares da disciplina “Linguagens Formais, Autômatos e Computabilidade” da turma 2023.2 do curso de Bacharelado em Ciência da Computação da Universidade Federal do Pará.

# 1. Introdução

[Falar brevemente sobre a hierarquia das linguagens]

[Falar brevemente sobre gramática, alfabeto e vocabulário]

[Falar brevemente sobre linguagens regulares]

[Falar brevemente sobre as regras de produção em linguagens regulares]

[Falar sobre o questionário – descrevê-lo]

# 2. Materiais e Métodos

Para implementar o código fonte que satisfaz os requisitos do questionário foi utilizada a linguagem de programação Python versão 3.10. O código fonte foi escrito utilizando os softwares Visual Studio Code e Pycharm em sistemas operacionais Linux usando as distribuições Ubuntu 22.04.2 LTS, Manjaro 22.1 e Fedora 37, bem como em sistemas operacionais Windows 10 e Windows 11. Para conferir suporte à operações com expressões regulares utilizou-se a biblioteca Re, que faz parte da biblioteca padrão da linguagem de programação escolhida, mas não vem habilitada por padrão. Para conferir suporte a anotações de tipo para o tipo função foi utilizada a classe Callable da biblioteca typing. Para criar testes automatizados para o código fonte foi utilizada a biblioteca pytest.

Para fazer o controle de versionamento do código foi utilizada a ferramenta Git, em especial através da plataforma GitHub. Da mesma plataforma também foi utilizada a ferramenta Projects para organizar as atividades de cada membro do grupo, onde utilizamos a ferramenta de tickets para manter o controle sobre a lista de afazeres e o progresso na completação da atividade.

## 2.1. Considerações gerais sobre o código fonte

O código fonte foi dividido em módulos para separar a lógica de cada questão de sua implementação concreta, que é feita no arquivo main.py no módulo pai. A estrutura completa está ilustrada na Figura 1.

[[figura 1: output do comando tree na raiz do projeto]]

Obedecemos os princípios do desenvolvimento baseado em testes, então começamos na etapa “red” do ciclo “red-green-refactor”, escrevemos uma série de testes base de acordo com os exemplos da questão, em seguida implementados as expressões regulares para solucionar os problemas de modo a obter êxito nos testes base entrando na etapa “green”, e depois refatoramos o código

A instrumentalização da resolução das questões foi inspirada pelo padrão de projeto comportamental conhecido como Strategy, onde utilizamos funcões de ordem superior para criar funções de primeira ordem para implementar as estratégias, traduzidas nos reconhecedores de cada linguagem do questionário. O uso dessas tecnicas funcionais afasta a implementação aqui apresentada da aplicação padrão deste padrão de projeto que utiliza classes abstratas. Essa escolha foi feita para simplificar a legibilidade do código e para tirar proveito das capacidades da linguagem Python que permite essa abordagem funcional, inclusive através de decoradores.

O construtor dos reconhecedores está localizado no arquivo helper.py dentro do módulo common/, e funciona conforme o excerto apresentado no Algoritmo 1.

[[Algoritmo 1]]

Isso permite que possamos criar reconhecedores no código fonte munidos apenas da expressão regular que descreve a linguagem. Além disso, ao adotar essa estrutura, conferimos coesão ao código, uma vez que sabemos que cada reconhecedor vai ter o mesmo comportamento padrão, e, ao mesmo tempo geramos um grau de desacoplamento que nos permite refatorar cada reconhecedor individualmente onde a sua estratégia foi definida.

# 3. Detalhamento sobre os reconhecedores

Cada questão do relatório exigia a criação de um reconhecedor para uma dada linguagem. A seguir são explicadas as expressões regulares geradas para descrever esses reconhecedores.

## 3.1 Primeira questão

As mascaras de validação requisitadas na primeira questão foram assim implementadas.

[q1.nome]

[q1.email]

[q1.senha]

[q1.cpf]

[q1.telefone]

[q1.datetime]

[q1.numero]

## 3.2 Segunda questão

Os reconhecedores para os arranhos familiares apresentados foram assim implementados.

[q2.a]

[q2.b]

[q2.c]

[q2.d]

[q2.e]

[q2.f]

[q2.g]

# 4. Testes Experimentais

Para estressar os reconhecedores, primeiramente configuramos um sistema de testes unitários automatizados utilizando a biblioteca pytest. Os casos básicos de teste foram aqueles fornecidos no próprio questionário. Assim, um teste foi criado para cada exemplo em cada quesito apresentado na questão 1. Um exemplo de teste pode ser observado no Algoritmo 2.

[[Algoritmo 2]]

Os testes básicos para os reconhecedores necessários para resolver a questão 2 foram escritos manualmente.

Após a cobertura dos casos básicos, foram desenvolvidas estratégias para gerar casos adicionais. Para cada exigência das linguages apresentadas no questionário foram desenvolvidas funções — que funcionam como estratégias — para gerar sentenças compatíveis e cadeias incompatíveis. Essas estratégias alimentam uma função de ordem superior que retorna uma função de primeira ordem que por sua vez gera uma sentença ou cadeia semi-aleatória de tamanho variável de acordo com o padrão desejado.

Um exemplo desta implementação pode ser observado no Algoritmo 3.

[[Algoritmo 3: certificar de que o código apresentado gera uma cadeia de exemplo]]

Valendo-se desta ferramenta foram geradas funções extras para completar os testes básicos, certificando-se de que cada linguagem estava coberta por pelo menos vinte testes diferentes, de maneira a tentar cobrir os aspectos relevantes de cada linguagem. Quando a biblioteca pytest passou a sinalizar todos os testes como bem-sucedidos (Anexo 1) o código fonte foi considerado pronto e as questões resolvidas.

# 5. Comentários Finais

# 2. First Page

The first page must display the paper title, the name and address of the authors, the abstract in English and “resumo” in Portuguese (“resumos” are required only for papers written in Portuguese). The title must be centered over the whole page, in 16 point boldface font and with 12 points of space before itself. Author names must be centered in 12 point font, bold, all of them disposed in the same line, separated by commas and with 12 points of space after the title. Addresses must be centered in 12 point font, also with 12 points of space after the authors’ names. E-mail addresses should be written using font Courier New, 10 point nominal size, with 6 points of space before and 6 points of space after.

The abstract and “resumo” (if is the case) must be in 12 point Times font, indented 0.8cm on both sides. The word **Abstract** and **Resumo**, should be written in boldface and must precede the text.

# 3. CD-ROMs and Printed Proceedings

In some conferences, the papers are published on CD-ROM while only the abstract is published in the printed Proceedings. In this case, authors are invited to prepare two final versions of the paper. One, complete, to be published on the CD and the other, containing only the first page, with abstract and “resumo” (for papers in Portuguese).

# 4. Sections and Paragraphs

Section titles must be in boldface, 13pt, flush left. There should be an extra 12 pt of space before each title. Section numbering is optional. The first paragraph of each section should not be indented, while the first lines of subsequent paragraphs should be indented by 1.27 cm.

## 4.1. Subsections

The subsection titles must be in boldface, 12pt, flush left.

# 5. Figures and Captions

Figure and table captions should be centered if less than one line (Figure 1), otherwise justified and indented by 0.8cm on both margins, as shown in Figure 2. The caption font must be Helvetica, 10 point, boldface, with 6 points of space before and after each caption.



Figure . A typical figure



Figure 2. This figure is an example of a figure caption taking more than one line and justified considering margins mentioned in Section 5.

In tables, try to avoid the use of colored or shaded backgrounds, and avoid thick, doubled, or unnecessary framing lines. When reporting empirical data, do not use more decimal digits than warranted by their precision and reproducibility. Table caption must be placed before the table (see Table 1) and the font used must also be Helvetica, 10 point, boldface, with 6 points of space before and after each caption.

Table 1. Variables to be considered on the evaluation of interaction techniques



# 6. Images

All images and illustrations should be in black-and-white, or gray tones, excepting for the papers that will be electronically available (on CD-ROMs, internet, etc.). The image resolution on paper should be about 600 dpi for black-and-white images, and 150-300 dpi for grayscale images. Do not include images with excessive resolution, as they may take hours to print, without any visible difference in the result.

# 7. References

Bibliographic references must be unambiguous and uniform. We recommend giving the author names references in brackets, e.g. [Knuth 1984], [Boulic and Renault 1991]; or dates in parentheses, e.g. Knuth (1984), Smith and Jones (1999).

The references must be listed using 12 point font size, with 6 points of space before each reference. The first line of each reference should not be indented, while the subsequent should be indented by 0.5 cm.

# References

Boulic, R. and Renault, O. (1991) “3D Hierarchies for Animation”, In: New Trends in Animation and Visualization, Edited by Nadia Magnenat-Thalmann and Daniel Thalmann, John Wiley & Sons ltd., England.

Dyer, S., Martin, J. and Zulauf, J. (1995) “Motion Capture White Paper”, <http://reality.sgi.com/employees/jam_sb/mocap/MoCapWP_v2.0.html>, December.

Holton, M. and Alexander, S. (1995) “Soft Cellular Modeling: A Technique for the Simulation of Non-rigid Materials”, Computer Graphics: Developments in Virtual Environments, R. A. Earnshaw and J. A. Vince, England, Academic Press Ltd., p. 449-460.

Knuth, D. E. (1984), The TeXbook, Addison Wesley, 15th edition.

Smith, A. and Jones, B. (1999). On the complexity of computing. In *Advances in Computer Science*, pages 555–566. Publishing Press.