Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Arara: editor, interpretador e juiz on-line de código em português

William Coelho da Silva

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Orientador
Prof. Dr. Daniel de Paula Porto

Brasília
2023

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Arara: editor, interpretador e juiz on-line de código em português

William Coelho da Silva

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Prof. Dr. Daniel de Paula Porto (Orientador)
CIC/UnB

Prof. Dr. Guilherme Novaes Ramos
CIC/UnB

Prof. Dr. Edison Ishikawa
CIC/UnB

Prof. Dr. Marcelo Grandi Mandelli
Coordenador do Bacharelado em Ciência da Computação

Brasília, 19 de dezembro de 2023

Dedicatória

Eu dedico este trabalho à minha família por sua presença, união e especialmente dedicação que houve por parte da minha mãe, sra. Eloides Coelho de Sá, a grande responsável por me guiar nas minhas escolhas com sua vivência e sabedoria.

Agradecimentos

Agradeço ao IEEE por me fornecer artigos e suas respectivas citações de maneira descomplicada, agradeço ao Google Acadêmico por me ajudar a buscar os melhores artigos afim de ganhar embasamento e especialmente agradeço o meu orientador neste trabalho, Prof. Dr. Daniel de Paula Porto, por suas correções precisas e atenciosas, sua ótima didática, sua grande criatividade em escrever emails, suas boas ideias e especialmente por sua disponibilidade em período integral.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES), por meio do Acesso ao Portal de Periódicos.

Resumo

Contexto: A aprendizagem de linguagens de programação utiliza quase que exclusivamente a língua inglesa. Isso pode criar dificuldades iniciais para alunos que não têm domínio deste idioma. Além disso, os alunos iniciantes apresentam dificuldade no processo de instalação de ambientes de programação, compiladores/interpretadores e saber se os seus programas estão corretos ou não. **Objetivo**: Criar uma linguagem inteiramente em português com interpretador próprio e avaliação de resposta (juiz on-line). E que esse ambiente possua um visual moderno e possa ser acessado totalmente online sem a necessidade de instalar ou configurar algo. **Método**: Pesquisar ferramentas existentes neste contexto. Isso inclui linguagem de programação em português, plataformas online para programação em português, avaliador de código para estas plataformas. Além disso, foi necessário identificar e estudar linguagens de programação consolidadas no mercado e avaliar sua estrutura e conjunto de instruções que sirva de base para a criação de uma linguagem útil para a aprendizagem. Depois de fazer essa análise, a ferramenta foi criada e foi realizado um estudo piloto com a ela. No estudo piloto, foram coletados dados iniciais de feedback dos alunos dos primeiros semestres da universidade dos cursos de informática. **Resultados**: A ferramenta obteve bons resultados na avaliação dos alunos. No entanto, foram relatados alguns bugs e limitações relacionadas à interpretação da linguagem. **Conclusão**: Esta ferramenta tem potencial para ser útil na aprendizagem de conceitos de programação em escolas e universidades de todo o Brasil, mas exige algumas correções e melhorias para isso.

Palavras-chave: Linguagem de programação, interpretador, juiz online, português, editor de código on-line

Abstract

Background: Learning programming languages almost exclusively uses the English language. This situation can create initial difficulties for students who need to be proficient in this language. In addition, beginners need help installing programming environments, compilers/interpreters, and knowing whether their programs are correct. **Objective**: To create a language entirely in Portuguese with its interpreter and response evaluation (online judge). This environment has a modern look and can be accessed online without needing to install or configure anything. **Method**: The first step was to research existing tools in this context. This research includes programming language in Portuguese, online platforms for programming in Portuguese, and code evaluators for these platforms. In addition, it was necessary to identify and study consolidated programming languages on the market and evaluate their structure and instructions to serve as a basis for creating a valuable language for learning. After this analysis, the tool was created, and a pilot study was carried out. In the pilot study, initial feedback data was collected from students in the initial semesters of the university computer science courses. **Results**: The tool obtained good results in the student's evaluation. However, some students reported some bugs and limitations related to language interpretation. **Conclusion**: This tool has the potential to be helpful in learning programming concepts in schools and universities throughout Brazil, but it requires some corrections and improvements to do so.

Keywords: Programming language, interpreter, online judge, portuguese, on-line code editor

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

CIC Departamento de Ciência da Computação.

ENIAC Electronic Numerical Integrator and Computer.

FGA Faculdade do Gama.

IDE Ambiente de Desenvolvimento Integrado.

JRE Java Runtime Environment.

MVC Model, View, Controller.

UCB Universidade Católica de Brasília.

UnB Universidade de Brasília.

Capítulo 1

Introdução

1.1 Contexto

Com o passar do tempo é possível observar a criação de diversas novas ferramentas na área da computação, seja ela produzida para ser utilizada nas áreas mais básicas até as mais complexas das necessidades humanas e com a educação não é diferente. É frequente que as pessoas sejam apresentadas à algo para auxiliar em sua aprendizagem, desde áreas mais gerais até as mais específicas e desconhecidas pela maior parte da sociedade.

O processo de aprendizagem de linguagens de programação pode espantar algumas pessoas que nunca tiveram contato com tal tecnologia previamente. Durante o curso de ciência da computação na Universidade de Brasília, pude perceber que alguns fatores são notáveis para isso ocorrer. Um deles é que boa parte das linguagens de programação utilizadas no mercado, e, por consequência ensinadas, utilizam termos em língua inglesa. Isso decorre da própria história da computação, pois seus principais expoentes no desenvolvimento nos primórdios eram britânicos como Charles Babbage, Ada Lovelace e Alan Turing e americanos como John Presper Eckert e John Mauchly [1].

O Brasil é um país cuja a população possui baixo conhecimento de língua inglesa. Em média, apenas cerca 5% das pessoas fala inglês, sendo 1% fluente [2]. Para aprender os conceitos de programação, saber falar inglês não é algo fundamental, tampouco ser fluente. Entretanto, saber ler e entender algumas palavras-chave é uma condição sem a qual não pode ser realizada a programação de um software.

[1] Brunelli Moreno,J **A história do ENIAC, o primeiro computador do mundo** https://tecnoblog.net/especiais/eniac-primeiro-computador-do-mundo-completa-65-anos/, acesso em 2023-09-07.

[2] Loureço, Aline **Apenas 5% da população brasileira fala inglês, aponta pesquisa** https://www.segs.com.br/educacao/347834-apenas-5-da-populacao-brasileira-fala-ingles-aponta-pesquisa, acesso em 2023-09-07

É importante salientar que o aluno com baixo nível de inglês pode decorar tais palavras-chave utilizadas na programação e suas funções, porém, terá grandes dificuldades ao compreender mensagens de erro, na deputação de software e na leitura de documentações de materiais programação [3].

Outro ponto importante é a necessidade de instalação de compiladores/interpretadores de linguagem de programação em computador pessoal e de um editor de código ou Ambiente de Desenvolvimento Integrado (IDE). Além disso, pode ser necessário inserir comandos para compilar e executar o programa, todos esses passos podem causar um certo incômodo para os menos experientes no assunto.

Estas pessoas poderiam, com o tempo, se envolver com a programação e mudar as suas vidas por adentrarem em um mercado que cresce muito a cada ano [3] [4], porém possuem dificuldade de dar o primeiro passo.

Existem diversas ferramentas que auxiliam a aprendizagem de linguagens de programação [4], porém poucas se destacam por fornecerem este serviço utilizando termos totalmente em português e em uma plataforma on-line.

Tendo isso em mente foi elaborada uma proposta de fornecer gratuitamente uma ferramenta on-line que seja capaz de apresentar as funcionalidades básicas em português de uma linguagem de programação utilizada no mercado. Contudo, durante as pesquisas para a realização deste trabalho, foi identificado que já existem alguns serviços gratuitos que podem ser utilizados para esse mesmo propósito. Levando este aspecto em consideração, foi elaborada um proposta de criar uma ferramenta que, além de dar o suporte ao aprendizado de algoritmos, possuísse também uma funcionalidade de avaliador de respostas para validação de tarefas, conhecido no meio da computação como "juiz on-line".

Esta proposta inicial possui um processo que engloba:

- Cadastro de atividade por parte do professor;
- Após o cadastro da atividade, um link será gerado para que os alunos possam realizar a tarefa;
- O juiz online testará automaticamente o programa submetido pelos alunos com as entradas e saídas previamente cadastradas pelo professor;
- Um página de acompanhamento das tarefas realizas por si mesmo ou pelos os alunos.

[3] ORTIZ, S. M.; ENRIQUEZ, D. **Previsões da IDC apontam crescimento de 5% do mercado de TIC no Brasil em 2023**. Disponível em:<https://www.idc.com/getdoc.jsp?containerId=prLA50352423>. Acesso em: 1 out. 2023.

[4] DAU, G. **Mercado tecnológico nacional apresentou crescimento de 118% nos últimos dez anos**. Disponível em: <https://www.jornalcontabil.com.br/mercado-tecnologico-nacional-apresentou-crescimento-de-118-nos-ultimos-dez-anos/>. Acesso em: 1 out. 2024.

1.2 Motivação

Cada vez mais a necessidade de formar profissionais de TI se torna necessária para atender a alta demanda na área ao redor do mundo e também no Brasil [5]. E para isso o ensino da programação necessita abranger um grande número de pessoas, sempre se aprimorando e procurando novas maneiras de atrair e manter novos estudantes da tecnologia.

Como citado anteriormente, um possível dificultador para se aprender programação é que, no geral, as linguagens comerciais e boa parte das utilizada didaticamente serem em inglês. Isso afasta especialmente aqueles que não tiveram o ensino do inglês básico durante a sua formação na educação básico, normalmente, os mais pobres, uma vez que o Brasil amarga a 54ª posição no ranking Pisa [6].

Este ponto pode ser provado com base no estudo de Theresa Beaubouef e John Mason (2005) [5], professores da *Southeastern Louisiana University* relataram que nas universidades que trabalharam anteriormente eram aconselhados a ensinar uma linguagem de programação no lugar de uma língua estrangeira pois aprender uma nova língua é difícil. Tal afirmação pode ser transposta para o contexto deste projeto, isto é, em um país com déficit tão acentuado na capacidade de utilizar o inglês, a aprendizagem de uma língua estrangeira pode ser mostrar mais complicada que de uma linguagem de programação.

Ainda sobre o ponto da linguagem, é interessante de se examinar o estudo *More Time or Better Tools? A Large-Scale Retrospective Comparison of Pedagogical Approaches to Teach Programming* [6]. Nele os autores citam que linguagens como Python, por possuírem sintaxes simples, são melhores para o aprendizado. Nesse caso, a sintaxe diz respeito aos comandos simples que realizam tarefas interessantes como ordenar uma lista, ler algo, ou armazenar valores em uma variável. Diz respeito também à forma de escrita da linguagem que se assemelha ao inglês falado. Seria razoável então, por inferência, sustentar que uma linguagem de programação em português auxiliaria na aprendizagem de programação.

Outro ponto que pode auxiliar na aprendizagem de linguagem de programação é a prática de exercícios e documentação da linguagem. O artigo *Analysis of the teaching-learning methodology adopted in the introduction to computer science classes* [7] mostra que 69% dos alunos entrevistados na Faculdade do Gama (FGA) desaprovam o material

[5]JATOBÁ, M. **Estudo revela que falta mão de obra qualificada no setor de tecnologia**. Disponível em:<https://www.folhape.com.br/ECONOMIA/2373-ESTUDO-REVELA-QUE-FALTA-MAO-OBRA-QUALIFICADA-SETOR-TECNOLOGIA/128753/>. Acesso em: 2 out. 2023.

[6]Maia, Rodrigo: **Educação brasileira está em último lugar em ranking de competitividade.** https://www.cnnbrasil.com.br/nacional/educacao-brasileira-esta-em-ultimo-lugar-em-ranking-de-competitividade, acesso em 2023-06-18

didático deixado pelos professores e 57% relatam que a falta de exercícios práticos são um fator negativo ao aprender programação.

A possibilidade da prática de exercícios e a sua submissão para um avaliador automático auxilia no processo de aprendizagem de linguagens de programação bem como pode auxiliar o professor no gerenciamento da turma na aplicação de tarefas e coleta de resultados [8].

Vale ressaltar que, além da própria linguagem de programação mais acessível, cadastro de tarefas e corretor automático, o ambiente onde será realizado tudo isso é tão importante quanto. Existem diversas opções no mercado de Ambiente de Desenvolvimento Integrado (IDE) a editores de código mais simples, porém, grande parte deles e, em especial os mais conhecidos, foram feitos para o indústria de software e não para aprendizagem de programação [9].

A instalação de uma Ambiente de Desenvolvimento Integrado (IDE) e a configuração do ambiente para programar como compiladores, interpretadores, variáveis de ambiente, etc. gera dificuldades relatadas pelos alunos [10], portanto, a criação de um ambiente totalmente configurado para qualquer máquina acessá-lo pelo navegador de internet removeria esse dificultador.

Dito isso, é possível dizer que as motivações deste projeto são os seguintes pontos:

- Inserir novas pessoas no mundo da tecnologia de maneira gradual utilizando palavras já conhecidas por falantes de português;
- Facilitar a aprendizagem de programação para todos, especialmente por alunos da Universidade de Brasília (UnB);
- Criar um ambiente completo para aprendizagem e avaliação;
- Criar uma linguagem de programação possuidora de componentes que comprovadamente facilitem a aprendizagem de conceitos de programação;
- Otimizar a aplicação de atividades e feedback de correção.
- Ambiente para exercitar conhecimentos sem qualquer tipo de configuração;
- Deixar uma ferramenta que pode ser aprimorada livremente pela comunidade;
- Aproximar professor e aluno em um ambiente livre.

1.3 Objetivos

1.3.1 Objetivo Geral

O principal objetivo deste trabalho é o de oferecer uma nova ferramenta chamada Arara que minimize as principais dificuldades apresentadas no início da aprendizagem de programação. Tal proposta envolve a criação de uma linguagem de programação própria, totalmente em português, tendo como base outras linguagens de programação, estudadas na Seção 2.5, sejam elas linguagens comerciais ou focadas na educação.

A proposta envolve também a implementação de um Ambiente de Desenvolvimento Integrado (IDE) on-line totalmente configurado que execute o código escrito no editor de código de maneira simples, utilizando apenas palavras na língua portuguesa e dependente apenas da instalação de um navegador e conexão com a internet.

Para finalizar, é proposto também um sistema de avaliação onde o professor possa cadastrar tarefas com entradas e saídas esperadas e compartilhar a atividade com os seus alunos. Esses, por sua vez podem submeter um código, que será testado e terá um feedback do resultado da avaliação. Como complemento disso, também existirá uma tela de acompanhamento de resultados das submissões de quem realizou a tarefa.

1.3.2 Objetivos Específicos

O para atingir o objetivo educacional proposto, é necessário cumprir os seguintes objetivos específicos:

- Estudo comparativo entre várias linguagens de programação para identificar características que poderiam ser benéficas para o aprendizado dos alunos;
- Construção de uma linguagem de programação própria totalmente em português;
- Construção de um compilador e uma IDE em ambiente WEB, de forma que os estudantes não precisem instalar qualquer componente além de um navegador de internet;
- Documentação da linguagem criada de forma que os alunos possam aprender sua sintaxe na própria ferramenta;
- Construção de um juiz on-line capaz de apoiar o professor na parte avaliativa dos alunos;
- Avaliação básica inicial da ferramenta com um grupo de estudantes para verificar sua usabilidade e funcionalidades.

1.4 Metodologia de pesquisa

Este trabalho foi planejado para ser executado da seguinte forma:

- **Escolha do tema de pesquisa** Em conversa com o orientador deste trabalho para definição de um tema para o trabalho de graduação foi feita uma proposta para um projeto voltado a oferecer uma alternativa educacional.
- **Prototipação e criação de uma prova de conceito da ferramenta Arara** No momento de apresentação da proposta foi mostrado a ferramenta Calango [11] para ser tomada como uma fonte de inspiração inicial e com importantes modificações e atualizações, como fazer ser inteiramente acessível pela WEB.

  Para a prova de conceito, foram levantados todos os requisitos iniciais do protótipo, sendo eles: editor de código, interpretador para linguagem própria, construção de telas, cadastro de usuário e de tarefa, banco de dados ferramentas para implementação.

  Foi feito um projeto simulador de comportamento real para testar funcionalidades sem necessariamente implementá-las por completo. Este objeto é chamado de *mockup*. Isto foi realizado com o propósito de verificar as hipóteses levantadas. Com este teste foi possível afirmar que o projeto poderia ser construído no prazo proposto de dois semestres.
- **Análise comparativa das principais linguagens de mercado**

  Foi feita uma análise comparativa a afim de colher informações e melhores caminhos a se trilhar no processo de desenvolvimento da linguagem de programação Arara.

  É possível dividir em dois grupos as linguagens estudadas:

  - **Linguagens comerciais:** linguagens bastante utilizadas e consolidadas no mercado.
  - **Linguagens educacionais:** linguagens que são utilizas para ensinar conceitos de programação e implementações básicas.
- **Definição da linguagem**

  Foi definida uma linguagem exclusivamente em português, utilizando *tokens* compactos para facilitar memorização e que remetessem aos *tokens* mais utilizados nas linguagem consagradas do mercado, que utilizasse a forma mais usual de definição de escopo, utilizando { para iniciar um novo escopo e } para fechá-lo.

  A ideia que o software Arara ensine conceitos de programação diversos e facilitasse a aprendizagem para vários tipos de linguagens fez com que fosse adotado uma

tipagem forte, além dos operadores lógicos e aritméticos simbólicos (||, &&, +, -, etc.). A escolha da tipagem forte se dá por conta da linguagem Arara se basear em alguns aspectos na linguagem C, o motivo desta escolha será detalhado na Seção 2.6.6.

- **Implementação das funcionalidades**

  O Arara foi desenvolvido utilizando a linguagem de programação JavaScript aplicada ao *framework* ReactJS. Entre os componentes deste software estão:

  - **Planejamento e versionamento** - Os requisitos levantados na parte da prototipação foram desmembrados em diversas tarefas anotadas na plataforma de gerenciamento de projetos Trello e atualizada conforme a necessidade de desenvolvimento, bem como a comunicação contínua entre orientador e aluno para desenvolvimento, esclarecimento, melhorias e relatos de bugs. Para manter registro foi utilizada a tecnologia de controle de versão Git por meio da plataforma GitHub. Essa plataforma também provê a hospedagem do site Arara atualmente.
  - **Interpretador** - O interpretador foi implementado inteiramente em JavaScript. Funciona em duas etapas, na primeira é feita a coleta do código escrito pelo usuário no editor de código e verificando se há erros léxicos, sintáticos e semânticos, já na segunda parte é feita a execução efetiva do código também como a possibilidade de detecção de alguns erros.
  - **Juiz on-line** - O Juiz on-line possui muitas semelhanças com o interpretador, com o diferencial que as entradas não são colhidas do teclado do usuário e saídas não são mostrada na tela. As entradas cadastradas são inseridas automaticamente pelo juiz durante a execução do código e suas saídas são comparadas às do cadastro da atividade.
  - **Parte visual** - Totalmente desenvolvida com o já citado *framework* ReactJS. A opção nesta parte foi por facilidade de acesso e por um visual moderno e adotando temas escuros por conta do conforto visual.
  - **Banco de dados** - Como o Arara não possui comunicação com um servidor, ou seja, todo o processamento de dados é feito no próprio *front-end*, é necessário a utilização de um banco de dados hospedado em nuvem. Dado isso, foi decidido por utilizar o Firebase Firestore Database por ser um serviço gratuito, confiável, e com uma ótima documentação.

- **Avaliação da ferramenta por alunos**

Após o exercício proposto em sala de aula pelo professor/orientador, foi feita a coleta por meio de dois canais diferentes:

- Para coleta de dados geral foi elaborada uma página para o feedback no software Arara.
- Para uma coleta de dados mais estruturada foi elaborado um formulário com perguntas selecionadas em conjunto com o orientador deste trabalho.

1.5 Organização

Este documento está organizado em capítulos, que por sua vez está organizado em seções. Os capítulos estão organizados da seguinte forma:

- Capítulo 1 - Introdução - Onde é apresentado os argumentos que levaram a elaborar este projeto e os objetivos gerais esperados;
- Capítulo 2 - Revisão Literária - Levantamento e explicação referenciada de funcionalidades importantes relacionadas com este projeto;
- Capítulo 3 - Ferramenta - Descrição de todos os componentes que compõe o projeto desenvolvido e seus métodos de funcionamento;
- Capítulo 4 - Estudo Piloto - Apresenta o estudo aplicado à alunos da Universidade de Brasília (UnB) e os resultados obtidos;
- Capítulo 5 - Considerações finais - Onde é apresentado os resultados gerais do projeto, o que pode melhorar e no que trabalhar posteriormente.

Capítulo 2

Revisão Literária

2.1 Considerações inciais

O propósito deste capítulo é o de fazer uma revisão literária afim oferecer um arcabouço para o entendimento e explicação de decisões tomadas durante o desenvolvimento do projeto Arara. Este capítulo está organizado de maneira que seja possível compreender teoricamente os principais componentes do projeto.

2.2 Métodos de tradução

A tradução é um processo que pode ser definido como converter a linguagem da mensagem sem que o sentido se altere [1]. O processo de tradução é um dos principais componentes que estão envolvidos na área de desenvolvimento de software.

Uma linguagem é uma forma de passar uma mensagem [2], como a língua portuguesa, inglesa, Língua Brasileira de Sinais (Libras) ou até mesmo gestos utilizados no cotidiano como o positivo e balançar a cabeça. A computação não foge à regra. Há diversas linguagens de programação que são formas diferentes de se escrever uma mensagem.

Como pode ser visto no ranking de linguagens de programação mais utilizadas no ano de 2023 [1], todas elas são legíveis a um humano, mesmo tendo que ser treinado para tal. Porém, o computador como é conhecido atualmente não entende a linguagem humana. Portanto é preciso conseguir transmitir o teor do que foi escrito por uma pessoa para "passos" que a máquina consiga executar, ou seja, uma linguagem que a CPU (*Central Processing Unit*) entenda. É neste ponto onde entra o compilador.

[1] Bastos, Athena: **Linguagens de programacao mais usadas**. https://www.alura.com.br/empresas/artigos/linguagens-de-programacao-mais-usadas, acesso em 2023-07-06.

Há diferentes formas de se traduzir um algoritmo. São elas: a compilação e interpretação [2]. Classicamente ambas têm as suas nuances e especificidades, porém atualmente esta divisão não é tão clara assim e ambas as maneiras de tradução absorveram pontos positivos uma da outra. Mas de qualquer maneira, ainda há pontos a serem destacados entre elas.

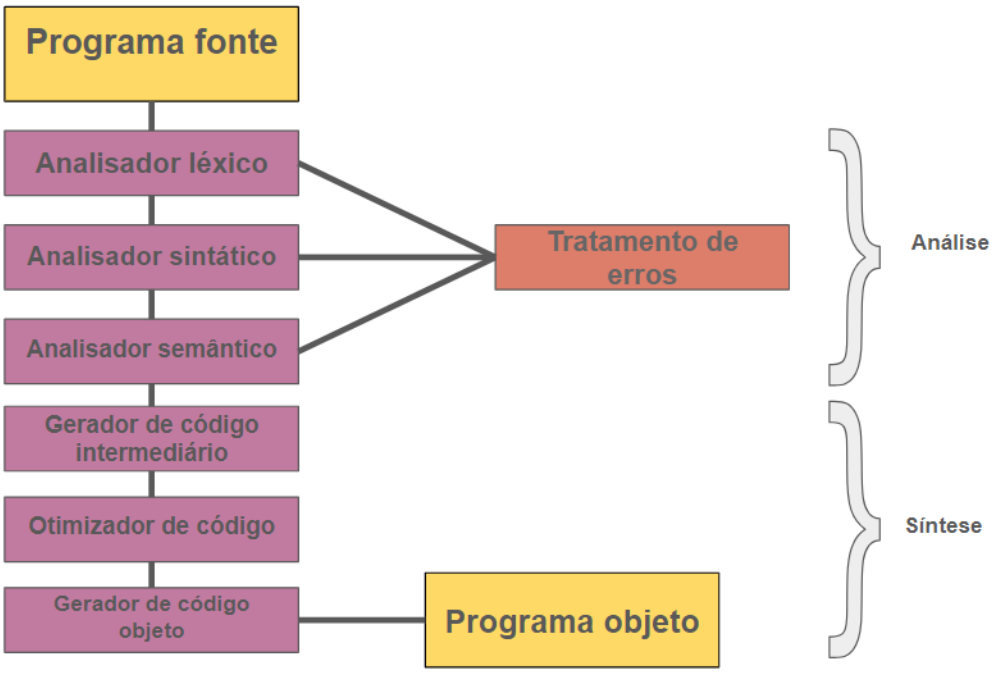


Figura 2.1: Passos envolvidos no processo de tradução de um programa [1]

2.2.1 Fases de tradução

A tradução pode ser dividido em duas fases: análise e síntese. Enquanto a fase de análise tem como principal atribuição a de encontrar erros e tratá-los (caso seja possível) ou avisar ao programador o problema que ocorreu, a fase de síntese tem como objetivo a construção do programa objeto escrito na linguagem alvo [1]. Para a construção deste programa objeto é necessário ter um programa fonte (código-fonte). Isto é, um algoritmo escrito em uma linguagem de programação que será submetido a um tradutor afim de realizar algum procedimento.

Como dito anteriormente, mesmo havendo diferenças entre compilador e interpretador, há convergência no *modus operandi* destes dois tipos de tradutor que serão detalhados nas seções abaixo.

2.2.2 Analisador léxico

O termo léxico pode ser definido com a reunião/conjunto de todas as palavras existentes em uma língua[2]. Se tais palavras forem utilizadas em um contexto, estruturadas e organizadas de uma maneira correta, isto faz com que elas possuam sentido e possam expressar sentimentos, histórias, contos, dentre outras informações.

Na programação não é muito diferente. Existem diversas palavras utilizadas para dar comandos ao computador, intermediadas pelo tradutor. Tais palavras podem descrever comandos que são conhecidos e como pode-se nomear variáveis, funções, procedimentos, etc.

Reconhecido no meio dos tradutores como *scanner*, o analisador léxico tem a função de identificar tais palavras reservadas ou possíveis de serem declaradas. Estas são nomeadas como *tokens* [1].

Caso existam erros léxicos, o *scanner* tem o papel de detectá-los nesta fase. Os erros léxicos podem ser generalizados como um *token* que não pertence àquela gramática, como uma palavra reservada escrita errada, ou o nome de uma variável escrita de uma maneira não aceita.

Além da função de reconhecimento de identificadores e palavras-chave, é possível atribuir ao *scanner* o papel de remover comentários e espaços em brancos desnecessários, função que também pode ser atribuída ao pré-processador. Isso se torna necessário para evitar problemas nas próximas etapas de análise já que espaços em branco desnecessários acarretariam em um dificuldade maior na confecção dos demais analisadores.

O trabalho de gerar constantes é dado ao *scanner*, isto é uma operação totalmente dependente de operadores declarados já no código do programa. Ou seja, independente de valores externos e possivelmente variados, é razoável então que o analisador léxico transforme o resultado dessa operação em uma constante para se poupar trabalho de adicionar mais *tokens*. Isso acontece uma vez que cada elemento isolado (números, cadeia de caracteres, etc.) vira uma unidade autônoma o que deixa o processo de tradução desnecessariamente mais complexo [1].

2.2.3 Analisador sintático

O analisador sintático, chamado de *parser* é iniciado a partir do resultado do *scanner* para se aproveitar de algumas facilidades possibilitadas por ele. Pode-se definir o *parser* como um validador de sentenças de acordo com as regras gramaticais da linguagem fonte, ele é o encarregado de reconhecer e validar comandos como: declaração, expressões aritméticas, condicionais, laços de repetições, etc.

[2]Léxico: Dicionário Dicio. https://www.dicio.com.br/lexico/, acesso em 2023-09-26

É bastante comum que *parsers* gerem uma estrutura fruto da análise do código fonte seguindo as regras gramaticais da linguagem. A estrutura mais utilizada é a estrutura de árvore, chamada de árvore gramatical ou árvore de derivação [1].

Um ponto importante nesta etapa é que esta árvore não é uma derivação completa do código fonte e sim uma versão já otimizada. Para isso são tomadas algumas providências como a eliminação de redundâncias por exemplo.

Algo a ser observado como um diferencial entre *scanner* e *parser* é a forma de trabalho deles. Enquanto o analisador na verificação de identificadores e palavras-chaves se vale de uma correspondência à uma expressão regular, conhecido no meio da computação como *match in regex*, sem qualquer tipo de recursão nesse caminho, o analisador sintático é baseado na recursividade para validação de sentenças.

É possível definir uma lógica geral para geração desta árvore gramatical, como o código em Python descrito abaixo que terá cada elemento catalogado com nomes clássicos (não necessariamente sempre utilizados) e formarão ramos dessa árvore.

```
valorFinal = salarioDia * 22 - descontos
```

1. **valorFinal** Identificador de variável
2. = Sinal de atribuição
3. **salarioDia** Identificador de variável
4. ***** Sinal de multiplicação
5. **22** Valor literal numérico
6. **-** Sinal de subtração
7. **descontos** Identificador de variável

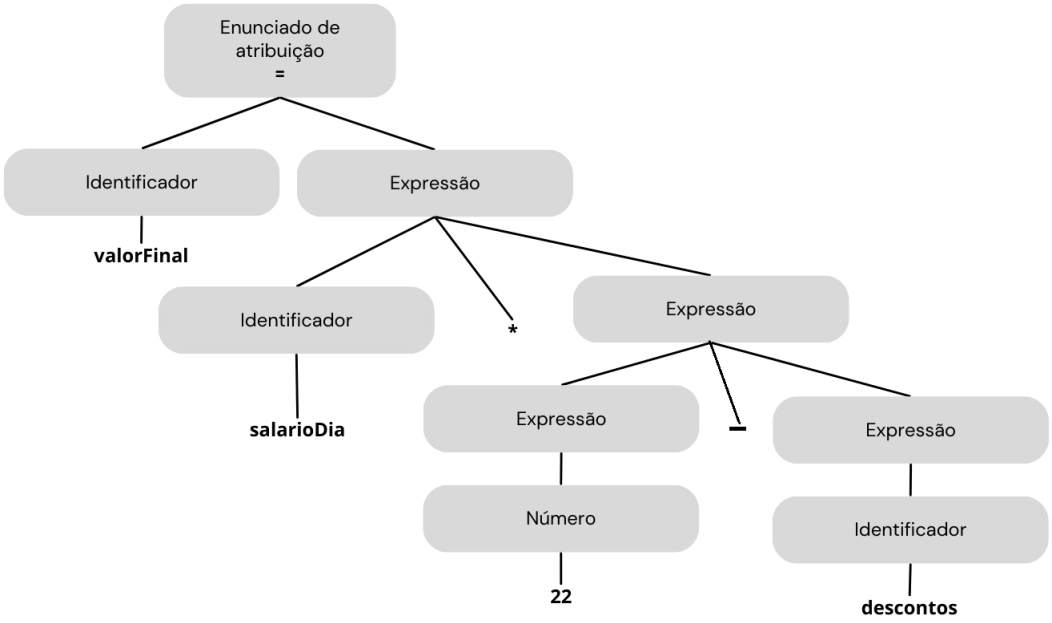


Figura 2.2: Representação visual da árvore gramatical gerada pelo tradutor que será utilizada em outros passos

2.2.4 Analisador semântico

A principal atribuição do analisador semântico é o de utilizar a árvore gerada no *parser* para poder verificar erros de tipagem. Isso pode ser exemplificado como quando uma atribuição de um número inteiro é feita à uma variável do tipo texto, isso deve gerar um erro e quem se encarrega disso é o próprio analisador semântico [1].

Existem indicações de problemas na programação que são chamados de *warnings* ou avisos em português. Os *warning* não são necessariamente erros, mas sim mensagens de diagnósticos que indicam que algo está errado[3]. O analisador semântico geralmente lida com esse tipo de problema também, como quando é feita uma atribuição real à uma variável de um tipo inteiro. Algumas linguagens permitem isso e o analisador semântico pode se encarregar disso para fazer a conversão de um tipo para outro.

2.2.5 Gerador de código intermediário

O código intermediário é o primeiro processo de síntese após a fase de análise. Geralmente o código intermediário é feito com base no código de três endereços [1]. Ou seja, o código-fonte será traduzido para uma linguagem que não é linguagem alvo ainda.

[3]GNU Compiler Collection: **Gcc warning options**, 2023-12-29. https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html, 2023-12-29.

O código de três endereços deve possuir duas propriedades fundamentais: ser fácil de produzir e fácil de traduzir para a linguagem alvo. Este tipo de código é pensado para ter no máximo três operandos, um operador e uma atribuição [1]. O principal objetivo para se gerar este tipo de código é o de facilitar a otimização pois os otimizadores podem se especificar bastante no aprimoramento de apenas um código e não grande quantidade possível de linguagens fonte.

2.2.6 Otimizador de código

Após a geração do código intermediário é feita uma otimização que pode para reduzir o tempo de execução do código. Esta otimização possui diversas especificidades, mas é possível exemplificar como o código abaixo, como demostrado no livro *Compilers: Principles, Techniques and Tools* [1]:

```
// transforma o número 60 em real e armazena em temp1
temp1 := intToReal(60)
temp2 := id3 * temp1
temp3 := id2 + temp2
id1 := temp3
```

Tal código pode ser transformado no código abaixo sem qualquer impacto no sentido original.

```
temp1 := id3 * 60.0
id1 := id2 * temp1
```

2.2.7 Gerador de código

É o ponto onde o código intermediário é traduzido para um código de máquina realocável, ou seja, um código que a máquina onde está sendo compilado entende (definido pelo tipo de processador que a maquina utiliza) e pode ser executado em qualquer ponto da memória.

2.2.8 Compilador

É possível generalizar e afirmar que o compilador segue todos os passos citados acima nos processos de tradução. No caso de linguagens compiladas o programa precisa ser

compilado por inteiro antes de ser executado [4]. Exemplos de linguagens compiladas são o C, C++ e Pascal.

2.2.9 Interpretador

Como feito com o compilador, é possível generalizar e afirmar que interpretadores cumprem os passos citados acima, mas as linguagens interpretadas possuem a característica de serem executadas de sentença em sentença [5]. Exemplos de linguagens interpretadas PHP, Perl e Python.

2.2.10 Semelhanças e diferenças

Como dito anteriormente, atualmente a divisão entre linguagens compiladas e interpretadas não é tão mais clara. É possível tomar o Java como exemplo.

A linguagem Java é compilada e interpretada. Os programas escritos em Java são compilados em um formato intermediário chamado *bytecode* pelo seu compilador javac. Depois este código intermediário é utilizado pelo interpretador Java Runtime Environment (JRE) [12].

Mas ainda assim é possível destacar diferenças entre estes dois tipos de tradução na Tabela 2.1.

2.3 Ambiente de Desenvolvimento Integrado (IDE)

A sigla IDE é comumente utilizado no cotidiano em um ambiente de programação, ela se refere ao conjunto de ferramentas comuns utilizadas no desenvolvimento em uma interface gráfica afim de facilitar certos aspectos do trabalho de um programador[6].

Editor de código é a funcionalidade mais básica que um software pode ter para ser utilizado na escrita de código. Porém IDEs possuem ferramentas que vão além disso, como [13]:

- Compilação e detecção de erros no próprio editor, ou seja, integração com o compilador ou interpretador da linguagem;
- Navegação para definições de classes, rotinas e variáveis com base no clique;

[4]Guide, Ionos Digital: **Compilers vs. interpreters**. https://www.ionos.com/digitalguide/websites/web-development/compilers-vs-interpreters/, acesso em 2023-10-07

[5]Guide, Ionos Digital: **Compilers vs. interpreters**. https://www.ionos.com/digitalguide/websites/web-development/compilers-vs-interpreters/, acesso em 2023-10-07

[6]RED HAT. **O que é IDE? - Ambiente de Desenvolvimento Integrado**. Disponível em: <https://www.redhat.com/pt-br/topics/middleware/what-is-ide>. Acesso em: 5 jan. 2024.

Tabela 2.1: Comparativo entre tipos de tradução [2]

Compilador	Interpretador
A tradução do programa acontece durante a compilação	A tradução do programa acontece durante o tempo de execução, ou seja, o código é interpretado linha por linha.
O código de máquina gerado é armazenado em um disco secundário como executável.	Como a execução ocorre imediatamente após a tradução do programa, o código da máquina é armazenado temporariamente na RAM.
Como os compiladores podem realizar otimizações em tempo de compilação, seu desempenho é significativamente mais rápido do que a linguagem interpretada.	Como os intérpretes traduzem o código durante o tempo de execução, há muito pouco espaço para otimização, pois o processo de tradução incorre em penalidades de desempenho que resultam em desempenho mais lento.
As linguagens compiladas são mais adequadas para aplicativos críticos de desempenho, como programas utilitários para um sistema operacional.	Linguagens interpretadas são mais adequadas para automação de tarefas e scripts.

- Transformações ou refatorações automáticas de código;
- Busca e substituição em um grupo de arquivos;
- Possibilidade de desfazer em vários níveis;
- Formatação específica para a linguagem atual;
- Ajuda interativa para a linguagem em edição, ou seja, mostrar sugestões relevantes para complementar o código.

2.4 Juízes eletrônicos

A aprendizagem de programação é um processo que evolui com a prática, isto envolve, entre outras coisas, elaborar programas e conferir se os seus resultados estão conforme esperado. Fazer este tipo de checagem manualmente é algo custoso e passível de erro [14].

Juízes Eletrônicos são, de forma geral, programas que avaliam outros programas automaticamente [15]. Para que isso ocorra é necessário que o avaliador tenha em sua base de dados as entradas e saídas esperadas para um determinado programa que será avaliado.

De maneira geral, estes avaliadores usariam este conjunto de entradas cadastradas e utilizariam nos programas a serem avaliados, com isto eles compararam a saída do programa com a sua respectiva entrada e verificaria sua exatidão deste programa [14].

2.5 Linguagens de programação

É possível definir linguagem como uma capacidade exclusiva do ser humano se expressar e também de transmitir e receber conhecimento por meio de um sistema carregado de simbolismos e sentidos [16]. Já as linguagens de programação podem se enquadrar como linguagens artificiais, tal como a matemática. Ou seja, trata-se de algo planejado para repassar instruções a um computador na forma de um algoritmo.

2.5.1 Histórico

O primeiro computador totalmente eletrônico da história foi o Electronic Numerical Integrator and Computer (ENIAC), encomendado pelo exército do Estados Unidos da América para ser utilizado na Segunda Guerra Mundial. Entretanto começou as suas operações apenas em 1946, após o término da guerra. Seu sistema era baseado em papéis perfurados [7].

Os primeiros computadores da década de 1940 eram programados modificando mecanicamente as rotas de energia em *Assembly* que é uma linguagem de baixo nível. Mesmo sendo de baixo nível precisa ser traduzida para o binário para que o computador possa executar [2].

É possível notar uma evolução nas linguagens de programação ao longo das décadas e algumas tendências. Isso pode ser visto pela Tabela 2.2.

2.5.2 JavaScript

A abordagem direta ao JavaScript neste documento é feita por ser a linguagem utilizada para programar o software Arara. Portanto, entender a sua história, objetivo e funcionalidades agregaria no processo de entendimento do projeto Arara. É importante salientar que o JavaScript não foi considerado no estudo das linguagens descritos na Seção 2.6.

O JavaScript foi lançado em 1995 pela empresa Sun Microsystems para descrever a linguagem utilizada pelo navegador Netscape, atualmente Mozilla [17]. A linguagem JavaScript é considerada uma linguagem web que forma a tríade juntamente com HTML

[7]Fernandes, Daniel: **Do ENIAC ao notebook: confira a evolução dos computadores nas últimas décadas**. https://www.cnnbrasil.com.br/tecnologia/do-eniac-ao-notebook-confira-a-evolucao-dos-computadores-nas-ultimas-decadas/, 2021

Tabela 2.2: Evolução das linguagens ao longo das décadas [2]

Década	**Linguagens**	**Tendências**
1950	Autocode, FORTRAN, LISP, ARGOL 58, COBOL	Surgimento das linguagens modernas e consideradas de alto nível com o objetivo de atingir grandes instituições como o COBOL muito utilizado em sistema financeiro até hoje e o FORTRAN largamente empregado na área acadêmica.
1960 e 1970	APL, Simula, BASIC, Pascal, C, Smalltalk, Prolog, ML, SQL	Criação de linguagens muito influentes e onde se iniciou os paradigmas de programação. [2]
1980	Ada, C++, Eifell, Perl	Surgimento da programação orientada a objetos
1990	Haskell, Python, Java, Ruby, Lua, JavaScript, PHP	O surgimento da internet impactou enormemente as linguangens de programação que se adaptaram - as criadas nessa década são mais utilizadas atualmente[8]

e CSS para programação web. O HTML é utilizado para especificar o conteúdo da página, o CSS serve para estilizar a página, e por fim o JavaScript é utilizada para definir o comportamento desta página. Trata-se de uma linguagem interpretada, não tipada, dinâmica e de alto nível. Ela é altamente utilizada de acordo com o paradigma funcional e orientado a objetos [17].

2.5.3 Paradigmas de programação

Há muitas maneiras de se programar um computador a depender da necessidade do usuário ou negócio direcionado. Quando esta maneira de se programar é baseada em uma teoria matemática ou em um conjunto coerente de princípios recebe o nome de paradigma de programação [18].

Cada paradigma possui suas próprias regras e princípios, porém, isto não significa que uma linguagem que adota um paradigma necessita que apenas seja aplicado aquele conjunto de regras quando para utilizá-la. Isto se deve ao fato de existirem linguagens multiparadigma que incorporam mais de um conjunto de regas e princípios, sou seja, mais de um paradigma. Alguns exemplos de paradigmas de programação são:[18]:

- **Programação Orientada a Objetos (POO)** - Este paradigma é focado na definição de objetos que possuem atributos e métodos. Objetos são instâncias de classes

que definem sua manipulação [19].

- **Programação Funcional** - É um paradigma de programação que enfatiza a avaliação de funções matemáticas e evita o uso de estados e mutações de dados como o Haskell e Miranda [20];
- **Programação Concorrente** - É a atividade de construir programas que possuem duas ou mais atividades acontecem simultaneamente. Podendo acontecer com cada processos de posse de um processador (núcleo) ou com mesmo disputando o pelo mesmo processador [21].
- **Programação Lógica** - É um paradigma de programação que se baseia na lógica matemática. Um exemplo muito conhecido de linguagem que utiliza este paradigma é a linguagem Prolog [18].

2.6 Estudo das linguagens de programação

Para uma formulação de software mais precisa, assertiva e bem posicionada foi feito um estudo das linguagens de programação. Neste estudo foram consideradas as linguagens de programação em português e as tradicionais em inglês. Além disso, foi considerado o ambiente em que elas podem ser utilizadas.

Para minimizar o risco de ser feito algo muito arrojado ou diferente do que já há foi levado em consideração diversos pontos de cada linguagem como: facilidade de uso, diferenciais, comandos, estrutura de código, gramática e operadores. A seguir são apresentadas as linguagens utilizadas nesse comparativo.

2.6.1 Calango

Este é um software desenvolvido por alunos da Universidade Católica de Brasília (UCB) e da Universidade de Brasília (UnB) [11]. É um programa feito em java que precisa ser rodado localmente utilizando a Java Runtime Environment (JRE) a partir de um arquivo .JAR baixado no computador do usuário.

Trata-se de uma linguagem fortemente tipada e só utiliza comandos em português. Há pelo menos três seções no código do arquivo: uma para indicar o nome da classe e outros dois para indicar início e fim da função principal. É possível criar mais funções, mas sempre dentro de uma única classe.

Um ponto que chama atenção nesta linguagem é a definição de escopo que usa palavras para iniciar e fechar uma condicional e não símbolos, como abre chaves e fecha chaves. Além disso é utilizado palavras (**e** e **ou**) em vez de símbolos para operadores lógicos.

A linguagem Calango possui outras funcionalidades interessantes integradas ao ambiente de desenvolvimento como por exemplo: a possibilidade de salvar o código escrito, imprimi-lo, aumentar e diminuir a fonte, indentação automática, executor e até mesmo um depurador para executar passo a passo. Além de tudo isso possui uma documentação sucinta e direta ao ponto de interesse.

Certamente dois pontos que foram considerados e repensados para o Arara foram a acessibilidade e o aspecto visual. A dificuldade de acessibilidade se dá por conta de ser necessário baixar um arquivo .JAR ter o *JRE* para poder executá-lo. Isso pode parecer simples para quem tem experiência em computação mas é um processo que pode se mostrar problemático no início da aprendizagem sobretudo por envolver a instalação e configuração prévia da máquina virtual Java. No que fiz respeito ao aspecto visual, o Calango utiliza a linguagem Java para construir as suas telas e isto deixa as telas com um aspecto antigo. Para o Arara será utilizado o *framework* React (escolha detalhada na Seção 2.7.2 para a construção de seu *front-end* uma vez que esta tecnologia permite que façamos toda a parte visual do site Arara de forma elegante e moderna.

2.6.2 Portugol Webstudio

Portugol Webstudio [22] é uma das plataformas on-line de portugol on-line mais utilizadas. Um grande ponto forte desta plataforma é que é necessário apenas ter um dispositivo com acesso à internet e um navegador. Ou seja, não é necessária a instalação de qualquer programa no computador e nem a instalação de *plugins*, como o Mapler [23] e o Portugol on-line [24], respectivamente.

Todos os comandos utilizados nesta linguagem são em português, utiliza uma estrutura semelhante à linguagem C para definição de escopos, declaração de variáveis, condicionais, operadores aritméticos e lógicos. Além disso possui um ótima documentação que aborda vários assuntos necessários de maneira sucinta para conhecimentos básicos. Isso é de suma importância quando se trata de aprendizagem de programação.

2.6.3 C

Para Bertolini e outros [2], o C é considerada uma linguagem de médio nível por permitir tarefas comuns de linguagens de baixo nível como manipulação de bits, bytes e acesso de memória e também conter elementos característicos de linguagens de alto nível o que deixa a linguagem bem versátil.

Criadora de tendências, C consolidou o uso de vários comandos, símbolos e estruturas citados na Subseção Portugol Webstudio. Portanto, é possível afirmar que C utiliza o padrão do mercado, já que ela mesma o criou.

Um ponto negativo para o aprendizado que vale a pena ser ressaltado em C é a complexidade de se fazer a entrada e saída de dados que hoje se mostra defasado. Além disso, de acordo com o estudo *A controlled experiment on Python vs C for an introductory programming course: Students' outcomes* [25], o uso da linguagem C demonstra problemas no início da aprendizagem de programação em relação a linguagens mais próximas da linguagem humana e com facilitadores como listas. Por outro lado isto não significa que a aprendizagem da linguagem C não impacte favoravelmente a aprendizagem do aluno ao longo da vida.

2.6.4 Python

Python é uma das linguagem de programação mais utilizadas no mundo atualmente[9] e todo esse uso se deve também a fácil sintaxe do Python que se aproxima muito da linguagem natural do ser humano. O Python é uma linguagem dinamicamente tipada e possui uma estrutura e comando semelhantes ao C, porém, é possível afirmar que se trata de algo mais "limpo".

Uma questão a ser considerada nesta linguagem é a sua obrigatoriedade de indentação, se outras linguagens utilizam símbolos para definição de escopo, o Python torna desejado a utilização da indentação para programar. Esse é um dos pontos que fazem ela ser essa linguagem mais "limpa" citado anteriormente, portanto se torna necessário a utilização de uma Ambiente de Desenvolvimento Integrado (IDE) que ajude na indentação e nos problemas que a utilização errada pode causar.

É preciso salientar ainda que a entrada e saída de dados do Python é fácil, especialmente na entrada em que é possível mostrar uma mensagem para o usuário (geralmente uma pergunta ou ação a ser executada) já no mesmo comando de entrada. Outros pontos positivos são que os operadores lógicos são palavras em vez de símbolos (**and** e **not**) e os operadores aritméticos seguem o padrão de mercado.

O já citado estudo *A controlled experiment on Python vs C for an introductory programming course: Students' outcomes* [25] vai além e afirma que o Python possui algumas vantagens na utilização em cursos introdutório em relação ao C. Um dos pontos observados no estudo foi uma menor taxa de reprovação. Os autores justificaram esse fenômeno pelo fato de o Python ter uma sintaxe mais simples, tipagem dinâmica, semânticas de listas mais simples e a não utilização de chamada por referência [25].

[9]Cass, Stephen: **The Top Programming Languages 2023 — spectrum.ieee.org**. https://spectrum.ieee.org/the-top-programming-languages-2023, 2023. [Accessed 02-12-2023

2.6.5 Kotlin

Criada em 2011, Kotlin é a linguagem mais nova analisada neste estudo. Ela nasceu para ser um aprimoramento do Java. Portanto, em suas principais qualidades estão as resoluções de situações vistas como problema em sua linguagem irmã, como a verbosidade e melhoria do paradigma funcional[10].

Trata-se de uma linguagem estaticamente tipada que conta com tipos **val** e **var** onde o val é imutável e var é mutável, ou seja, **var** pode ter seu valor alterado e **val** não pode ter seu valor alterado. Uma vez feita uma atribuição a uma variável, é possível redefiní-la apenas por um valor do mesmo tipo do inicial. As sentenças: while, do while, if, else seguem o padrão do C, o For possui uma estrutura mais compacta e pode ser considerado semelhante ao do Python.

2.6.6 Conclusões do estudo

Com o estudo das linguagens acima foi possível ver várias diferenças entre elas. Todas elas possuem pontos fortes, fracos e características específicas. Foi feita uma tabela de comparação lado a lado das características de cada linguagem juntamente com o mesmo código escrito nas cinco linguagens estudadas. A tabela e os códigos estão disponíveis no Apêndice A deste documento.

Analisando todas estas linguagens citadas acima e levando em consideração o ponto de vista educacional, foi pensado em uma solução que fosse direta e intuitiva. Foi pensado também que os conhecimentos adquiridos na utilização do software Arara se torne algo útil quando for o momento de migrar para um dos grandes *players* do mercado de linguagem de programação.

Portanto, foi decidido que o software Arara só terá, inicialmente, uma única seção delimitada pelas *tags*: **INICIO_PROGRAMA** e **FIM_PROGRAMA**. Esta decisão foi tomada para que o estudante olhe para o programa e já tenha bem definido na sua mente um escopo onde ele poderá escrever códigos. Nenhuma outra seção foi adicionada para que o código funcionasse como uma classe (ou uma única unidade de código) e para que não houvesse dificuldade logo na etapa inicial de convivência do ambiente.

Como visto nesta seção, as diversas linguagens estudadas têm as suas peculiaridades. Portanto, a decisão foi a de manter com referência a estrutura da linguagem C. Esta escolha se deve pela linguagem C ser robusta e consolidada no mercado. Por este motivo ela é utilizada para desenvolver softwares que necessitam destas características, como sistemas

[10]Hugo, Kewerson: **Por que o kotlin foi criado?**, Jun 2017. http://blog.triadworks.com.br/por-que-o-kotlin-existe. acesso em: 2023-08-02

operacionais por exemplo [26]. Além disso, foi utilizada o Calango como inspiração, a qual possui uma estrutura e comandos semelhantes ao C.

Portanto, o aluno teria bastante facilidade de fazer conexão com linguagens de uso real e também, por ventura o aluno possua conhecimento prévio de outras linguagens, o caminho reverso é válido. Dito isso, será utilizado uma sintaxe semelhante a C para definição de escopos, condicionais, definição sentenças, declaração de variável, atribuição e operações aritméticas.

Como citado acima, a entrada e saída da linguagem C se tornaria um problema caso fosse escolhida como fonte de inspiração. Portanto é mais apropriado fazer tais comando tendo como base o Python. Para os laços de repetição foi decidido tomar como base o que é feito no Kotlin.

Já o que diz respeito a como será nomeado os comando e tipos, foi decidido adotar algo semelhante ao que é feito no software Calango [11] e no Portugol WebStudio [22]. Porém serão realizadas algumas mudanças como o tipo **cadeia**, que se refere à uma cadeia de caracteres e será alterada para **texto** e alguns outros pontos mais específicos que serão detalhados posteriormente neste documento.

2.7 Aplicações Web

É possível definir aplicações Web como um programa de computador que utilize um navegador de internet como cliente. O cliente é programa que o usuário utiliza para executar o aplicativo[11].

O software Arara foi criado utilizando o *framework* React e por isso se torna relevante explicá-lo e algumas de suas tecnologias auxiliares.

2.7.1 Navegador

O ambiente onde será acessado o software Arara é um navegador web de forma on-line. Esta ferramenta pode ser definida como um software que interpreta informações vindas de arquivos e sites para que possam ser visualizada pelo usuário [27].

Para Murillo [27], os navegadores são formados por diversos componentes como o motor de busca, motor de renderização, interpretador de JavaScript, etc. Suas funções são:

[11]Nations, Daniel: **What Is a Web Application? — lifewire.com**. https://www.lifewire.com/what-is-a-web-application-3486637. [Accessed 08-10-2023]

- Motor de busca: Coordena as ações entre a interface do usuário e o motor de renderizações, isto é, busca informações com base nas requisições do usuário para ser mostradas na tela.
- Motor de renderização: É o responsável por mostrar o conteúdo solicitado pelo usuário. Para isso é analisado o código HTML e CSS da página para montar cada subcomponente resultado na tela.
- Interpretador de JavaScript: Responsável por executar o código JavaScript escrito pelo desenvolvedor da página. Este ponto trabalha em conjunto com o motor de renderização, uma vez que o JavaScript descreve o comportamento visual que o HTML e CSS podem ter.

2.7.2 Escolha de *framework* para desenvolvimento

A utilização de um *framework* para desenvolvimento web se torna muito interessante pois garante a reutilização de código, programação orientada a objetos, modularização, bibliotecas prontas para uso e foco na lógica do negócio[28].

Pode-se definir três principais concorrentes de desenvolvimento web com *framework* JavaScript, são eles: Angular, ReactJS e VueJS. Todos eles possuem pontos fortes e fracos em relação aos seus concorrentes.

O ReactJS foi escolhido para construir esta ferramenta. Isso se deve ao fato do desenvolvedor deste projeto já possuir experiência com este *framework*. Além da farramenta possuir a maior comunidade entre essas três citadas e, por consequência, problemas podem ser resolvidos de formas mais célere e há uma maior gama de bibliotecas a serem utilizadas [28].

2.7.3 ReactJS

Criada pelo Facebook e lançada em 2013, o ReactJS, ou apenas React, é um *framework* para JavaScript para construção de páginas WEB [29]. O React tem ganhado muito espaço no mercado principalmente pela rapidez de desenvolvimento. Isto pode ser atribuído por conta de seu *auto-reload* que renderiza apenas os componentes que foram modificados. Isto permite que a aplicação possa ser desenvolvida e testada em tempo real, sem a necessidade de subir novamente a aplicação [29].

Um outro ponto é a facilidade de desenvolvimento. O Angular, o principal concorrente do React, utiliza uma estrutura semelhante ao Model, View, Controller (MVC), coisa que o React deixa mais decentralizado [29]. Além disso, o React possui diversas bibliotecas

gratuitas que podem ser facilmente incorporadas no projeto afim de ganhar performance, usabilidade e elegância no desenvolvimento web.

Capítulo 3

Ferramenta

3.1 Considerações inciais

O propósito deste capítulo é o de detalhar todo os passos que envolveram o desenvolvimento do software Arara, bem como relatar as dificuldades encontradas durante o processo, ideias deixadas pelo caminho, objetivos traçados e consequências das escolhas tomadas ao longo de toda a construção deste projeto.

Como dito anteriormente, o objetivo fundamental deste software é o de facilitar a aprendizagem de programação sem que o estudante precise saber uma palavra em inglês sequer. Além disso, para tornar esse aprendizado algo realmente útil, foram utilizados comandos e estruturas existentes em linguagens e já consolidadas no mercado, como em linguagens estruturadas e procedurais.

3.2 Escolha do nome

A escolha do nome Arara se deve pelo fato de ter sido apresentado pelo orientador o software Calango [11] no momento da explicação da proposta deste projeto. Apesar da palavra **calango** fazer parte da linguagem coloquial para nomear pequenos lagartos, ela passa uma sensação de algo muito brasileiro, assim como a palavra **arara**. Assim foi feita a escolha por esse nome por manter a escolha por nomear como um animal e também dificultar a confusão de nomes uma vez não ter sido encontrada qualquer outra tecnologia na área da computação que possua este nome.

3.3 Definição de escopo e objetivos

Como introduzido na Seção 1 o objetivo deste software é o de construir um serviço totalmente on-line para interpretação de comandos em uma linguagem própria criada para

este projeto.

Com isso em mente, foi feita uma pesquisa para conhecer o que já havia nessa área. Como o estudo descrito na Seção 2.6, este ponto foi muito importante para adequação do software à um espaço educacional que possibilitasse a aprendizagem dos principais conceitos de programação.

A proposta inicial foi fazer algo semelhante ao software Calango [11], mas de forma totalmente on-line. Ou seja, uma plataforma on-line que se possa editar código em português e com um compilador próprio. Então foi feita uma pesquisa do que já existia nessa área e foram encontrados alguns software como Portugol WebStudio [22], Mapler [23] e Portugol on-line [24].

É possível definir o software Arara como um facilitador pelo seu aspecto de auxílio ao professor. Auxílio no sentido de permitir avaliar seus alunos com tarefas cadastradas por ele mesmo em uma linguagem que deve se assemelhar aos principais *players* do mercado de linguagens de programação. Auxílio também por seu amparo aos alunos que podem aprender em uma linguagem que usa termos que são utilizados por eles no dia a dia e estruturada na forma das linguagens mais conhecidas e demandadas do mundo.

Além de aspecto educacional, é preciso atingir alguns objetivos como a manutenibilidade e escalabilidade para que seja possível corrigir eventuais problemas encontrados, adicionar novas funcionalidades a depender da necessidade. Assim a ferramenta poderá atingir cada vez mais pessoas.

É preciso que a linguagem criada utilize termos que são facilmente compreensíveis por brasileiros e que guie os estudantes para uma mais fácil adaptação às linguagens de uso no mercado de trabalho. Para isso é necessário que o compilador desta linguagem de programação consiga traduzir corretamente os comandos digitados pelo usuário e também consiga executar diretamente no navegador onde está rodando a aplicação. Além disso ele deve capaz de detectar erros no código escrito pelo usuário e indique e ponto do código problemático para o estudante.

Outra característica importante é ter uma interface visualmente agradável e moderna com funcionamento simples de modo que forneça todos os recursos sem qualquer dificuldade para o aluno e professor. Este site deve conter uma área de edição de texto, documentação, um espaço para os usuários depositarem um *feedback* e uma página com informações do criador desta plataforma.

Além disso, com citado anteriormente, para a diferenciação deste projeto foi elaborada um proposta de um analisador de código para verificar a corretude. Para isso é necessário uma área de cadastro de tarefa que deve conter espaços para informações que serão lidas pelo aluno, como a descrição e um espaço para digitar entradas e saídas esperadas com a resolução deste problema. Este juiz on-line deve ser transparente para o aluno e ter uma

área para informar ao professor o resultado individual de cada aluno que realizou a tarefa.

Seguindo nessa linha, será possível que este projeto conte com um ciclo completo de avaliação para ser utilizado no contexto de sala de aula e funcionar como uma facilitador na aprendizagem de uma linguagem de programação.

3.4 Estrutura, símbolos e comandos da linguagem Arara

A linguagem Arara é uma linguagem totalmente em português criada para ser utilizada no software Arara juntamente com seu próprio interpretador.

É possível classificar a linguagem Arara como pertencente ao paradigma estruturado e pode ter a sua estrutura dita como semelhante a linguagem C. Ela pode ser considerada como uma linguagem fortemente tipada, e os seguinte tipos que são suportados pela linguagem Arara são apresentados na Tabela 3.1.

Tabela 3.1: Tipos suportados pelo Arara

Tipo	Valores possíveis	Observações
inteiro	Qualquer valor inteiro	N/A
real	Qualquer valor com ponto flutuante	Deve-se utilizar o símbolo . entre a parte inteira e parte fracionada do número
logico	verdadeiro ou falso	Possível de ser utilizado em comparações
caractere	Qualquer caractere UTF	Só pode conter um caractere e deve ser declarado entre aspas simples
texto	Cadeia de quaisquer caracteres	Tamanho infinitamente grande de caracteres e deve ser declarado entre aspas simples

Todo programa no Arara deve ser escrito dentro do escopo da função principal que é delimitado com o *token* #INICIO_PROGRAMA e finalizado com #FIM_PROGRAMA. Este escopo já é definido quando o usuário entra no editor de código para que não seja necessário o aluno escrevê-los recorrentemente. Além disso, o código padrão inserido vem com um comentário entre estas definições de início e fim para informar o usuário que o código tem que ser escrito entre estes *tokens*.

Qualquer sentença utilizada na linguagem Arara deve ser finalizada com o símbolo de ponto e vírgula (;). Esta decisão foi tomada para facilitar a identificação de sentenças por

parte do interpretador (e com isso evitar bugs no processo e execução) além desta forma de se finalizar sentenças ser utilizada por diversas linguagens como C e Java.

Desta maneira foram implementadas algumas instruções que foram julgadas como essenciais para aprendizagem dos principais conceitos de programação. Essas instruções estão descritas na Tabela 3.2.

Tabela 3.2: Instruções suportadas pelo Arara

Instruções	Descrição	Exemplo
escreva	Escreve no terminal texto e/ou valor de variável do Arara	`escreva('Olá, mundo!');` `escreva('Valor:  ', x);`
leia	Lê algo do teclado e salva na variável passada como parâmetro	`leia(x);`
se	Executa escopo de código com base na comparação de parâmetro	`se(x > 10) { //algo }`
senao se	Executa escopo de código com base na comparação de parâmetro e caso a condição **se/senao se** imediatamente superior tenha sido negativa	`se(x > 10) { //algo }` `senao se(x > 5) { //algo }`
senao	Executa escopo de código caso a condição **se/senao se** imediatamente superior tenha sido negativa	`se(x > 10) { //algo }` `senao { //algo }`

enquanto	Executa escopo de forma cíclica enquanto a condição passada como parâmetro for verdadeira. Há a necessidade se dar valor ao iterador e modificar o seu valor no escopo	`inteiro i = 0;`
		`enquanto(i < 10) {`
		`//algo`
		`i++;`
		`}`

para	Executa escopo de forma cíclica enquanto a condição passada como parâmetro for verdadeira. A declaração/inicialização de variável e iteração é feita na mesma instrução	`para(inteiro i = 0; i < 10;`
		`i++){`
		`//algo`
		`}`

3.4.1 Operadores lógicos e aritméticos

Além das instruções citadas acima, a linguagem Arara dá suporte às principais operações matemáticas, isto é: adição, subtração, multiplicação e divisão. Os *tokens* utilizados para referenciar estas operações são os convencionais utilizados pela maioria das linguagens: +, -, * e /.

As operações matemáticas no Arara podem ser realizadas com os tipos inteiro e real. É necessário que haja uma variável de destino que receberá o valor da operação e também dois operandos e um operador. Os operandos podem ser variáveis e/ou números literais da seguinte forma:

```
real x = 1.5;
real y = 1.5;
resultado = x - y;
```

```
inteiro x;
leia(x)
resultado = x * 10;
```

Existem três operações lógicas possíveis na linguagem Arara: conjunção, disjunção e negação. OS símbolos de operação lógicas seguem o padrão da linguagem C, ou seja, a conjunção é simbolizada por **&&**, a disjunção por || e a negação por **!**. Além destes, existem os símbolos de comparação, que são apresentados na Tabela 3.3.

Tabela 3.3: Operadores lógicos suportados pelo Arara

Símbolo	Nome	Observação
`>`	Maior que	Verdadeiro se o elemento da esquerda for maior que o da direita
`<`	Menor que	Verdadeiro se o elemento da esquerda for menor que o da direita
`>=`	Maior ou igual a	Verdadeiro se o elemento da esquerda for maior ou igual ao da direita
`<=`	Maior ou igual a	Verdadeiro se o elemento da esquerda for menor ou igual ao da direita
`==`	Igual a	Verdadeiro se o elemento da esquerda for igual ao da direita
`!=`	Diferente de	Verdadeiro se o elemento da esquerda for diferente ao da direita

É possível utilizá-los para parâmetros na comparação de desvios condicionais e laços de repetição. Portanto é possível de ser utilizado desta maneira:

```
logico lgc = verdadeiro;
se(lgc){
    // algo
}
```

3.4.2 Vetores

A linguagem Arara também dá suporte à utilização de vetores. Neste ponto é possível traçar um paralelo com o que é feito na linguagem Portugol WebStudio [22]. Desta maneira é possível destacar alguns pontos para a utilização de vetores na linguagem Arara:

- Os vetores são de um determinado tipo. Portanto todas as posições do vetor devem respeitar esta tipagem ou será lançado um erro;
- Na instrução de declaração é obrigatório a definição de tamanho do vetor;
- Não é necessário passar valores para inicialização do vetor na declaração, porém, caso seja feita, é preciso cobrir todas as posições do vetor;
- Caso não seja inicializado, todas as posições do vetor serão inicializadas com **null**;
- É possível de fazer atribuição indexada após a declaração.
- É possível recuperar o valor de uma posição do vetor utilizando um índex para acessar a posição;
- Operações matemáticas podem ser realizadas com acesso indexado.

É possível também utilizar vetores combinados com os laços de repetição, como no código abaixo:

```
inteiro x[9] = [11, 22, 33, 44, 55, 66, 77, 88, 99];
inteiro i = 0;

enquanto(i < 9){
    escreva(x[i]);
    escreva('\n');
    i++;
}
```

3.4.3 Limitações

No momento da escolha do tema foi levanta uma série de requisitos, os quais já foram retratados anteriormente na Seção 3.3. Com isso, foi feita uma priorização juntamente com o orientador deste projeto com a finalidade de cumprir prazos e implementar aquilo que foi julgado com mais prioritário e deixando os menos prioritários caso houvesse tempo hábil para ser implementado.

Foi feita essa priorização de componentes deste projeto como um todo, desde a linguagem até a plataforma Arara. Entre os recursos pensados mas não implementados estão:

- **Funções** - Por ser uma linguagem de aprendizagem de conceitos básicos de programação, os desvios incondicionais e a passagem de parâmetro acabam ficando em segundo plano, a linguagem Arara aceita comandos apenas na função principal, portanto não é possível declarar novas funções;
- **Objetos** - Não existe a possibilidade de declarar novos objetos na linguagem Arara, tudo deve ser feito utilizando os tipos genéricos que a linguagem dá suporte. Desta maneira é possível definir que a linguagem Arara como monoparadigma;
- **Não é uma API** - A utilização do interpretador e juiz da linguagem Arara não estão disponíveis para serem utilizados por outras plataformas e serviços;
- **Responsividade** - O visual da plataforma Arara é otimizado para o uso no navegador, portanto, há certos encaixes visuais que se mostram estranhos em dispositivos móveis.

3.5 Arquitetura

A proposta do software Arara é o de funcionar em apenas um módulo, ou seja, sem separação entre *front-end* e *back-end*. Tal decisão foi tomada por dois fatores principais.

- Hospedagem facilitada no GitHub Pages: Por conta desse projeto ter sido realizado por um estudante, há a disponibilidade gratuita de hospedagem de aplicações *front-end* no GitHub Pages, isso se torna fundamental para aplicação de testes, validação por parte do orientador e submissão ao alunos.
- Evitar falhas na comunicação: Não é incomum que ocorram falhas em comunicação entre o site e o servidor, portanto esta escolha visa a maior disponibilidade da ferramenta possível.

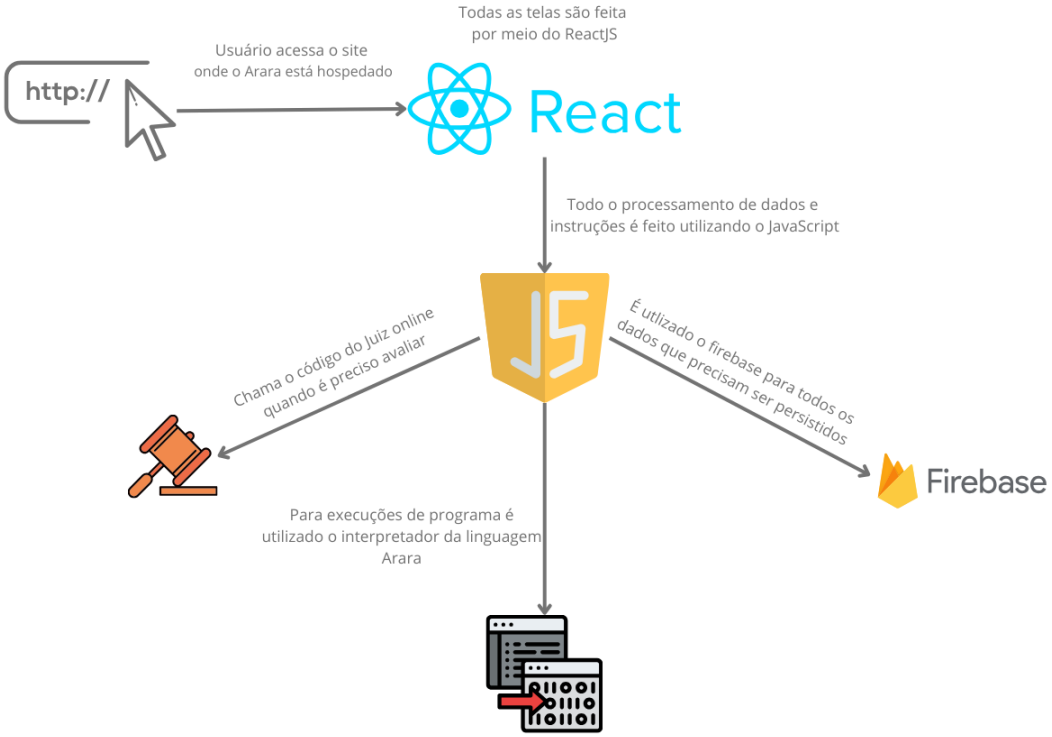


Figura 3.1: Principais componentes do Arara

3.5.1 Execução do programa

O principal fluxo de utilização do projeto Arara é o processo de execução de código e, conforme ilustrado na Figura 3.2, ele pode ser divido em três fases: escrita de código, interpretação/execução e entrada/saída.

Figura 3.2: Passos para a execução de um programa escrito no software Arara

A escrita de código é feita por meio da biblioteca de edição de código para React chamada Ace Editor. No momento que o usuário pressiona o botão de execução é feita a coleta do código em formato de texto e é iniciado o processo de interpretação do código.

Na Figura 3.2, o espaço que está intitulado como **interpretação** possui um subtítulo (analisador e executor). Isto indica a divisão do fluxo de interpretação do Arara, ou seja, o interpretador Arara possui duas etapas: análise e execução.

Foi divido desta maneira para melhor divisão de trabalho e reaproveitamento de código, uma vez que o fluxo de análise é idêntico para programas de teste(execução do usuário) e no fluxo de correção do juiz online. Ao passo que a fase de execução se diferencia para estes dois polos.

A etapa de análise do interpretador Arara é a primeira a ser realizada e ela pode ser divida em cinco passos principais:

- **Pré-processador** - Remove todos os comentários e espaços em branco (exceto entre aspas simples). Além de transformar o código escrito pelo usuário em um vetor de instruções com base no indicador de final de instrução obrigatório do Arara ";".
- **Analisador léxico** - Processa o resultado do pré-processamento e analisa os *tokens* utilizados em cada instrução.
- **Analisador sintático** - Após a validação dos *tokens* é feita validação de cada sentença. Nesta fase também é criada a tabela de uso e a tabela de definição.
- **Analisador semântico** - Realiza validações de tipagem, além de utilizar as duas tabelas citadas anteriormente para validar se as variáveis são utilizadas depois de declaradas.
- **Gerador de código executável** - Como resultado da fase de análise é construído um código um que possa ser interpretado pelo executor. Bem como é feita uma associação da condição e laço com o seu escopo.

O interpretador Arara é capaz de identificar os três tipos de erros já detalhados na Seção 2.2.

- **Erro léxico** - Ocorre quando é declarado um símbolo não especificado no dicionário da linguagem Arara. Todos os símbolos suportados estão especificados na Seção 3.4.
- **Erro sintático** - Quando há uma erro de sintaxe de alguma instrução reconhecida pelo dicionário da linguagem Arara.
- **Erro semântico** - Este tipo é lançado quando há algum problema de tipagem ou uso sem declarar. Ex: atribuir uma texto à uma variável declarada como inteira.

Além da identificação de erros, o interpretador Arara sempre tenta ajudar o máximo possível o usuário. Isto implica na necessidade de informar mensagens dos erros em português e, se possível, sugestões de como corrigi-lo como mostrado na Figura 3.3.

Note que que o usuário utilizou a instrução **escreva** de forma indevida. O *token* foi reconhecido mas ele é utilizado de forma errônea. Percebendo isso o interpretador lançou um erro descritivo para o usuário.

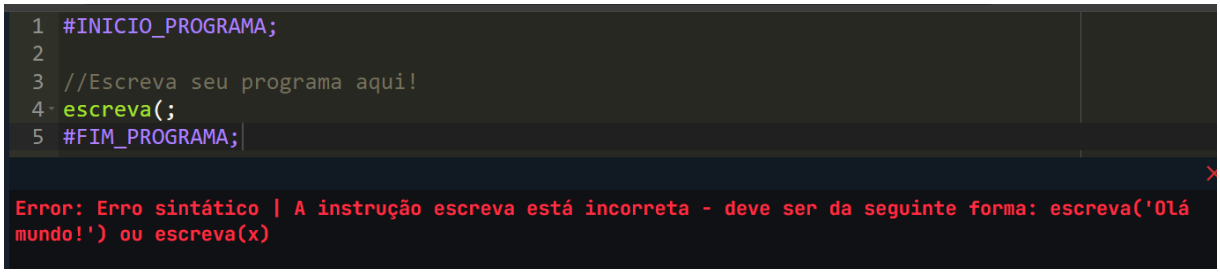


Figura 3.3: Exemplo de erro retornado pelo interpretador Arara

3.5.2 Validação de Respostas

Como dito anteriormente, há uma grande convergência entre a execução de usuário e a execução do juiz online, porém não em sua totalidade. Conforme ilustrado na Figura 3.4, a única mudança se dá na parte de entrada e saída no momento de execução do programa.

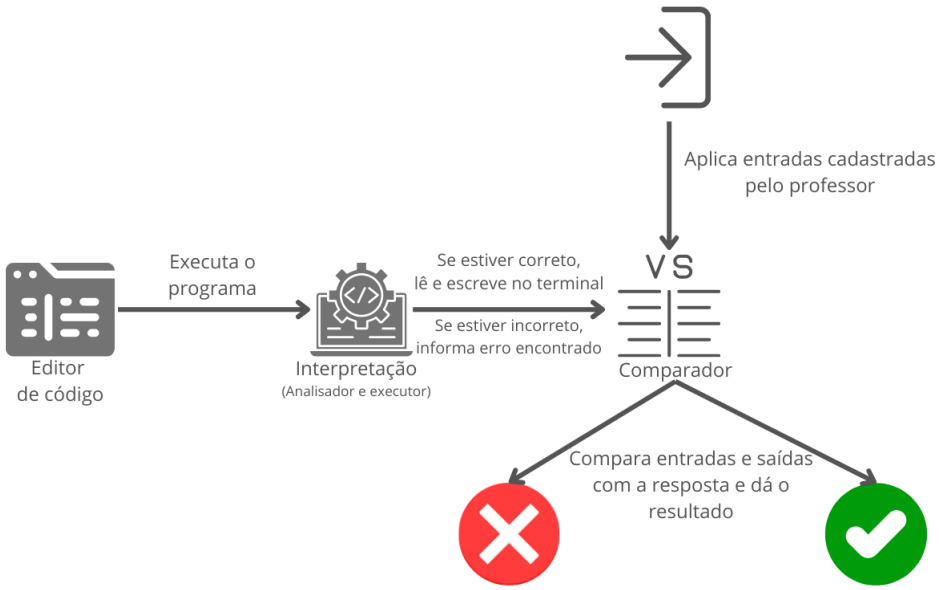


Figura 3.4: Ciclo realizado pelo juiz online para validação de atividades

Portanto, o juiz online da plataforma Arara substitui o fluxo de execução de usuário para receber entradas recuperadas do banco de dados (Firestore), que por sua vez foram cadastradas pelo professor no momento do registro da atividade. Ou seja, a instrução **leia** é substituída por uma atribuição direta à variável com o valor recuperado.

Seguindo essa linha, a instrução **escreva** também é alterada, em vez dela se comunicar com o terminal da plataforma Arara para visualização dos dados, o executor repassa esta informação para o juiz que por sua vez compara com a saída recuperada do banco de dados.

Além destas mudanças no fluxo de execução/validação, existe também uma adição final ao fluxo da atividade validada pelo juiz. Isto é, um salvamento de tentativa no banco de dados com o resultado associado ao usuário e à atividade. Isto serve para que o professor possa acompanhar a relação de alunos por atividade de forma individual.

3.5.3 Organização do banco de dados

O banco de dados desempenha função essencial na plataforma Arara. Esta ferramenta é utilizada para persistir dados que serão acessado posteriormente. O banco de dados Firestore (não relacional) está organizado em três coleções:

A primeira e mais simples delas é a coleção de *feedback*. Como todo software, o Arara não é a prova de falhas e esta coleção tem o papel de armazenar avaliações dos usuários caso eles a submetam. Possui apenas três campos: email, nome e mensagem.

Outra coleção é a de usuários. O seu principal objetivo é o de armazenar dados do usuário para possibilitar o login na plataforma. Mas esse não é o único, tem também o papel de salvar as tentativas do usuário de nas diversas atividades que ele realiza. Está organizado da seguinte maneira.

- **UID** - Tipo *string* - Identificador único do usuário;
- ***Name*** - Tipo *string* - Nome informado pelo usuário que será utilizado para exibição do cabeçalho quando estiver logado e também na tela de acompanhamento de atividades pelo professor;
- ***Email*** - Tipo *string* - Utilizado para poder fazer login;
- ***isTeacher*** - Tipo *booleano* - Dizer se o usuário é professor ou não. Isto limita o que é possível fazer sendo aluno;
- ***myAttempts*** - Objeto - Utilizado para saber se o usuário já realizou determinada atividade e o resultado dela. Está organizado da seguite maneira:
  - ***attemptCode*** - Tipo *string* - Identificador único da tentativa;
  - ***date*** - Tipo *date* - Data da tentativa;
  - ***name*** - Tipo *string* - Nome da tarefa;
  - ***sucess*** - Tipo *booleano* - Idicador de sucesso da tentativa;
  - ***detail*** - Tipo *number* - Porcentagem de acerto em comparação às entradas e saídas cadastradas pelo professor. Assume 100 se *sucess* for true;
  - ***taskCode*** - Tipo *string* - Identificar único da atividade na qual foi realizada a tentativa.

Por fim, a coleção de atividades fecha a lista. Este é o cerne da plataforma Arara. É utilizada em todo o escopo de validação de respostas e de cadastro de atividade. Ela é organizada desta forma:

- ***name*** - Tipo *string* - Nome da atividade;
- ***description*** - Tipo *string* - Descrição da atividade, ou seja, o que deve ser feito;
- ***qtdInput*** - Tipo *number* - Quantidade de entradas que o programa terá;
- ***inputFormat*** - Tipo *string* - Formato de entrada para instruir o aluno como ler as entradas;
- ***outputFormat*** - Tipo *string* - Formato de saída para instruir o aluno como escrever as saídas;
- ***outputExample*** - Tipo *string* - Exemplo de saída esperada;
- ***teacher*** - Tipo *string* - Identificador do professor que cadastrou a tarefa (user);
- ***tests*** - Tipo objeto - Armazena os testes aos quais a tentativa será submetida;
  - ***input*** - Tipo *array* - Vetor com entradas para a avaliação.
  - ***output*** - Tipo *string* - Texto com a resposta esperada da tentativa.
- ***attempts*** - Tipo objeto - Armazena dados que serão relevantes para acompanhamento pelo professor das submissões;
  - ***uid*** - Tipo *string* - Identificaor da tentativa;
  - ***date*** - Tipo *string* - Data da tentativa;
  - ***response*** - Tipo *Objeto* - Armazena informações da tentativa.
    - ***success*** - Tipo *booleano* - Resultado da avaliação;
    - ***detail*** - Tipo *string* - Porcentagem de acerto da tentativa;
    - ***results*** - [OPCIONAL] Tipo *array* - Apenas em caso de falha para que o professor possa ver detalhadamente cada entrada e saída do aluno e esperada, assim podendo saber onde exatamente ele errou.

3.6 Tecnologias

Para a realização deste trabalho foram empregadas uma série de tecnologias amplamente utilizadas no desenvolvimento de software. Elas foram escolhidas prezando pela gratuidade, disponibilidade e ampla documentação. Isto foi pensado para que o software

Arara possa desempenhar amplamente uma de suas funções que é ter um código aberto disponível para melhorias da comunidade.

O já citado Javascript foi combinado com o *framework* ReactJS para oferecer mais possibilidades e celeridade no desenvolvimento.

Para persistência das coleções citadas na Seção 3.5.3 é utilizado o Firestore. Esta é uma das muitas ferramentas oferecidas pelo Firebase/Google. Se trata de um banco de dados NoSQL, flexível e escalonável, como próprio nome já diz, é on-line, coisa que se torna necessário em uma aplicação que apenas roda em navegadores.

Este banco de dados armazena os dados em coleções no esquema chave-valor, estas coleções são contêineres de dados, além de dar suporte a vários tipos de dados de forma fácil e rápida.

Na parte visual, pode-se destacar as ferramentas Material UI e AceEditor. Estas bibliotecas para ReactJS foram de grande importância para o desenvolvimento deste projeto. O Material UI, ou MUI, é uma biblioteca para ReactJS que fornece um série de componentes visuais modernos, elegantes e personalizáveis que são utilizados no Arara, como: componente de *input*, componente de texto, botões, caixas de seleção, etc. Já o AceEditor fornece uma função essencial ao Arara, a possibilidade de ter um editor de código on-line com indentação e esquema de cores personalizado para a linguagem Arara.

3.7 Funcionalidades

O software Arara dispõe de um ciclo completo para aprendizagem de conceitos de programação e avaliação e isto implica na necessidade de implementação de diversas funcionalidades. As funcionalidades implementadas na ferramenta são descritas a seguir.

3.7.1 Cadastro de usuário

Para a utilização completa do software Arara é necessário o cadastro e login de usuário. Para o cadastro é necessário informar nome, email, senha e se é um professor ou não, este último ponto é necessário para definir o que o usuário terá acesso posteriormente. Todos estes dados são salvos no banco de dados Firebase Firestore.

3.7.2 Edição de código

No Arara pode-se editar código por meio da já citada biblioteca AceEditor. Com isto foi implementado um estilo de sintaxe, comumente conhecido como *Syntax Highlight*. Isto é de fundamental importância para que o aluno tenha um feedback instantâneo ao

escrever o código. Isto é, como monstrado na Figura 3.5, no momento da escrita do código o usuário poderá reconhecer se aquele é um *token reconhecido*.

```
#INICIO_PROGRAMA;

//Escreva seu programa aqui!
inteiro x;
leia(x);
x = x * 2;
escreva('O dobro é: ', x);

#FIM_PROGRAMA;
```

Figura 3.5: Exemplo da estilização com palavras-chave da linguagem Arara

3.7.3 Execução de código

O interpretador e executor da linguagem Arara já foi discutido na Seção 3.5.1.

3.7.4 Entrada e saída

Diferente do editor de código, o terminal utilizado pela saída foi criado desde o início sem a utilização de bibliotecas do JavaScript. Este componente exerce a função de se comunicar por meio de texto com o usuário, ele é capaz de escrever textos sempre que for utilizado um comando `escreva('...')` e ler do teclado do usuário com quando for codificado o comando `leia('...')`. Exemplos de entrada e saída de dados no terminal podem ser vistos na Figura 3.6.

```
10
O dobro é: 20

---------- PROGRAMA FINALIZADO ----------
```

Figura 3.6: Estado final logo após o usuário digital algum número.

No momento do clique para executar, o componente mantém o cursor piscando para indicar que espera que o usuário digite algo. Ao finalizar toda a entrada de dados, o componente mantém em tela o que o usuário digitou e escreve o resultado. Ao final escreve uma mensagem chamativa para indicar que o programa foi finalizado.

Este componente muito semelhante à um terminal de sistema operacional e é pensado para que o usuário se habitue com o uso de ferramentas deste tipo. Além disso o componente pode ocupar um tamanho personalizável na tela e também fechado sem a necessidade de cancelar a execução.

3.7.5 Cadastro de Atividade

O cadastro de atividade na plataforma Arara só está disponível para o professores registrados. Neste ponto foi pensado em uma tela que leve a entender o preenchimento em três passos. A figura 3.7 ilustra o seguintes passos.

- Informações gerais
  - Nome da tarefa;
  - Descrição da tarefa - Explicação do problema a ser resolvido;
- Formato de entrada e saída
  - Formato de entrada - Descreve como as entradas serão fornecidas e de que tipo ela será. Por exemplo: dois números inteiros;
  - Quantidade de entradas - O total entradas serão fornecidas a cada ciclo de execução do programa;
  - Formato de saída - Como é esperado que saída do programa será. Por exemplo: texto 'resultado' seguido do valor do cálculo;
  - Exemplo de entradas e saída - Exemplo direto para auxílio de atividade, ou seja, deve ser utilizado para indicar entradas e saídas esperadas para ajudar o aluno a visualizar o que deve ser feito.
- Entradas e saídas esperadas - Uma tabela que permite inserção de entradas e saídas para posterior avaliação.

Figura 3.7: Imagem da tela de cadastro de tarefa. Cada subseção detalhadas acima é visualmente dividida para o usuário

Após o cadastro da atividade, todos os dados inseridos nesta tela são gravados no banco de dados. Além disso, é gerado um link para que o professor possa compartilhar com os alunos que irão submeter códigos.

3.7.6 Avaliação de código

O aluno, ao acessar o link, é redirecionado ao editor de código descrito na Subseção 3.7.2. Entretanto, por se tratar de uma tarefa, existem algumas particularidades:

- *Pop-up* com as informações cadastradas na atividade pelo professor, exceto entradas e saídas esperadas;
- Botão de avaliação - No modo tarefa, é disponibilizado um botão para que o usuário possa submeter o seu código escrito no juiz online, bem como mantém o botão de execução para testes;
- Necessidade de login para que possa ser atribuída nota ao aluno que realizou a atividade e constar no painel de acompanhamento do professor.

Toda a avaliação de código é transparente para o usuário, ou seja, ele não precisa entender nada de como ela funciona, apenas fazer com que o seu programa resolva o problema e escreva a resposta esperada na tela.

3.7.7 Monitoramento de atividade

Logo após o cadastro da atividade será disponibilizada esta atividade no painel de acompanhamento do professor. Neste painel é possível acompanhar de forma isolada cada atividade cadastrada pelo professor. Já na subtela de cada atividade é possível ver todas as informações cadastradas descritas na Subseção 3.7.5.

Nesta subtela de atividades também é possível acompanhar cada aluno individualmente. Ou seja, é criado um campo mostrando as submissões que cada aluno, coma condição que ele tenha submetido ao menos uma vez, como pode ser visto na Figura 3.8. Além disso, o professor pode visualizar cada submissão individualmente também com suas saídas em cada um dos testes cadastrados. Isto é importante para que o professor possa identificar eventuais problemas no cadastro da atividade ou até mesmo em dificuldade dos alunos para determinadas tarefas.

Figura 3.8: Imagem da tela de Monitoramento de atividade. Mostrando individualmente cada aluno que submeteu uma tentativa.

3.7.8 Enviar feedback

Como se trata de um software muito recente, um dos pontos principais para melhorias e correções é ter a opinião do usuário. Para isto foi implementada uma simples tela

que conta com um único campo de texto em que o usuário pode escrever livremente sua opinião, relatar bugs, dar elogios, etc. Esta informação é salva no banco de dados e pode ser consulta posteriormente pelo desenvolvedor.

Capítulo 4

Pesquisa de Opinião

4.1 Considerações iniciais

Para ter uma aplicação realmente útil no ambiente de ensino de conceitos de programação, seria interessante colher a opinião dos alunos. Foi pensando nisso que foram formuladas formas para que fosse possível colher informações relevantes relativas ao Software Arara.

4.2 Métodos de avaliação

Foram elaboradas duas formas para para colher estas informações relacionadas à aplicação Arara. Tais formas serão detalhadas na subseções 4.2.1 e 4.2.2.

4.2.1 Feedback na plataforma Arara

Este item é referente à uma ferramenta própria e permanente da aplicação Arara, já explicada na Subseção 3.7.8. Este componente tem o objetivo de colher qualquer texto que o usuário logado submeter, ou seja, ele pode escrever livremente a sua crítica, comentário, elogio, etc.

Além de seu objetivo é importante ressaltar que estes dados só são acessíveis aos desenvolvedores que possuem a permissão de visualizar o banco de dados da aplicação. Isto garante segurança e também a privacidade do texto, uma vez que apenas a parte interessada terá acesso.

4.2.2 Formulário especializado

O formulário é uma forma mais organizada e planejada de se colher informações a respeito do uso da aplicação Arara, além de outros dados sobre conhecimentos dos alunos.

Além disso, não foi pedido autenticação para responder o formulário, isto foi feito desta maneira para não "constranger" o aluno a falar bem da ferramenta [30], ou seja, foi utilizado o anonimato como recurso para extrair respostas mais sinceras e evitar qualquer viés nesse sentido.

Outra preocupação foi a de não utilizar muitas perguntas ou então perguntas que necessitariam de respostas muito extensas. Portanto foi feito um corte apenas na perguntas mais interessantes e que pudessem fornecer respostas mais relevantes para adequação do Arara. Existia também a possibilidade de responder quantas questões o aluno desejasse e isto não o impediria de submeter o formulário.

4.2.3 Método de aplicação do formulário

Este formulário foi aplicado na forma de uma pesquisa de opinião na disciplina de Estrutura de Dados na turma de Engenharia da Computação da Universidade de Brasília (UnB) na qual o orientador deste projeto leciona. Os alunos participaram de uma sessão de implementação que durou uma hora e cinquenta minutos para conhecer a ferramenta Arara. Esta sessão ocorreu durante uma aula composta (dois horários seguidos) e contou com a presença do professor e do monitor da disciplina. O professor possuía um conhecimento geral da ferramenta por acompanhar a implementação, já o monitor da disciplina não a conhecia.

Durante esta seção os alunos foram convidados a implementar alguns código utilizando a ferramenta e pouco tempo depois do término da aula foram convidados voluntariamente a responde este formulário.

4.3 Perguntas realizadas

Foram formuladas oito perguntas de opinião aos alunos e aplicadas por meio da ferramenta *Google Forms*. Algumas era do tipo de múltipla escolha e outras discursivas. Todas elas são descritas na subseções a seguir.

- **Questão 1 - Em que ano e semestre você ingressou no curso?**

  O formulário foi iniciado com esta simples perguntas. Ela deveria ser respondida de forma discursiva. Uma vez que esta pergunta poderia ter muitas respostas diferentes, isto exigiria o cadastro de todas as combinações de ano e semestre em um tempo arbitrariamente julgado razoável.

- **Questão 2 - O quanto você sabia programar antes de ingressar no curso?**

Esta pergunta tem a finalidade de extrair uma média de conhecimento de uma turma de pessoas com diferentes percursos que ingressam em seus cursos na Universidade de Brasília (UnB).

Ela poderia ser respondida na forma de múltipla escolha. Foram dadas as seguintes opções de respostas e suas respectivas descrições:

  - **Nada**
  - **Pouco** - Conhecia alguns conceitos, mas nunca fiz nada elaborado;
  - **Razoável** - Entendia o principais conceitos e já fiz algoritmos que vão além do básico;
  - **Bem** - Já havia desenvolvido projetos com um alto grau de complexidade;
  - **Outros** - (Resposta livre)

- **Questão 3 - Qual era o seu nível de inglês antes de inciar o seu aprendizado em programação?**

  Aqui é onde é avaliado outro parâmetro muito relevante no contexto de aplicação da ferramenta Arara, uma vez que uma de suas principais áreas de atuação é justamente na dificuldade que os alunos brasileiros tem com a utilização de termos, instruções, ler documentação, etc. em inglês.

  Assim como a anterior, esta questão deveria ser respondida na forma de múltipla escolha.

  - **Nenhum** - Conhecia pouquíssimas palavras em inglês;
  - **Baixo** - Compreendia os termos mais utilizados mas não conseguiria ter uma conversa simples em inglês;
  - **Médio** - Conseguiria conversar temas básicos em inglês;
  - **Alto** - Conseguiria conversar assuntos diversos em inglês, além de ler e escrever textos.

- **Questão 4 - Qual foi a sua principal dificuldade na disciplina de Algoritmos e Programação de Computadores - APC?**

  Esta questão deveria ser respondida de forma discursiva e tem o objetivo de colher informações relevante em uma possível matéria de atuação da ferramenta Arara. Ou seja, poderia servir como uma base de informação do que poderia ser aplicada à linguagem ou plataforma Arara.

- **Questão 5 - Como a foi sua experiência com o software Arara?**

  Como esta pesquisa de opinião foi o primeira realizada com a ferramenta Arara, torna-se necessário colher informações a respeito da utilização inicial da ferramenta. Esta pergunta esperaria uma resposta mais ampla, isto é, saber como cada usuário achou da proposta, aplicação, linguagem, plataforma, etc. como um todo.

- **Questão 6 - O software Arara apresentou problemas? Caso positivo, quais?**

  Como todo software, o Arara pode precisar de ajustes, correções e atualizações. Ter a visão geral do usuário para este tipo de questão é essencial para correções futuras. Esta pergunta também deveria ser respondida de forma discursiva.

- **Questão 7 - Acredita que a ferramenta pode ser útil para aprendizagem de programação?**

  Esta questão é dedicada exclusivamente à opinião pessoal do aluno. É pedido que o aluno responda discursivamente se a ferramenta apresentada à eles seria útil. Mesmo sabendo que isso pode mudar de acordo com atualizações e correções da linguagem e ferramenta.

- **Questão 8 - Acredita que se tivesse começado a aprender programação utilizando o software Arara teria tido menos dificuldade no caminho?**

  Nesta pergunta o objetivo foi o de perguntar a percepção individual das pessoas, fazendo com que eles se colocassem em uma posição anterior no tempo. Cada aluno deveria pensar em si mesmo quando começou a aprender programação e relacionar este estado onde ele se encontrava com uma ferramenta que poderia ajudar ou não.

4.4 Respostas colhidas

4.4.1 Campo de feedback

No campo de feedback foram submetidos doze comentários. Apesar de ser um campo totalmente livre dentro da plataforma Arara para o usuário digitar o que quiser, é possível categorizar esses comentários em três grupos que serão detalhados a seguir.

- **Elogios**

  Dentre estes doze comentários, dois foram elogios parabenizando pela iniciativa e falando que a proposta é interessante e que pode ser útil de alguma maneira no ensino de programação.

- **Sugestão de melhoria**

  Outros três comentários recomendam alterações e melhorias na linguagem e plataforma Arara. Ex: declarar mais de uma variável na mesma linha, possibilidade de salvar e revisitar código por usuário.

- **Relato de bugs e dúvidas sobre comportamentos estranhos**

  Os outros sete comentários foram relatando bugs e dúvidas de porque determinado código (enviado no feedback) não funcionava. Todos os bugs relatados foram referentes à linguagem Arara e seu interpretador.

4.4.2 Formulário

No formulário aplicado à turma de Estrutura de Dados foram obtidas vinte e três respostas voluntárias. Pouco mais da metade da turma onde foi feita a avaliação respondeu à pesquisa de opinião. Além disso, pode-se adiantar que, mesmo não sendo obrigatório, todos os alunos que responderam ao formulário assinalaram todas as questões.

Os dados detalhados a seguir serão tratados como a opinião dos alunos da turma. É possível também visualizar as respostas completas por aluno no Apêndice B.

- **Resposta questão 1**

  A questão 1 perguntava em qual ano e semestre o aluno ingressou no curso. Observando as repostas é possível identificar que quase metade dos alunos que responderam estão no segundo semestre. Este é o semestre correto seguindo o fluxo para cursar esta disciplina nos cursos de Ciência da Computação e Engenharia da Computação. É importante destacar que, conforme observado na Figura 4.1, há muitos outros alunos espalhados em semestres anteriores.

- **Resposta questão 2**

  A segunda questão perguntava o nível de conhecimento de programação de cada aluno antes de ingressar no curso. Nesta questão pode-se observar que está bem divido nas três primeiras alternativas da questão. Com isso, nenhum aluno afirmou que teria um alto domínio de programação. Um gráfico com a porcentagem das respostas dos alunos pode ser visto na Figura 4.2.

  De forma mais precisa, dez (43,5%) alunos relataram que não sabiam programar antes de ingressar no curso. Isto representa quase metade dos que responderam ao formulário. O restante estão distribuídos de forma parelha: sete (30,4%) alunos com pouco conhecimento e seis (26,1%) com um conhecimento intermediário.

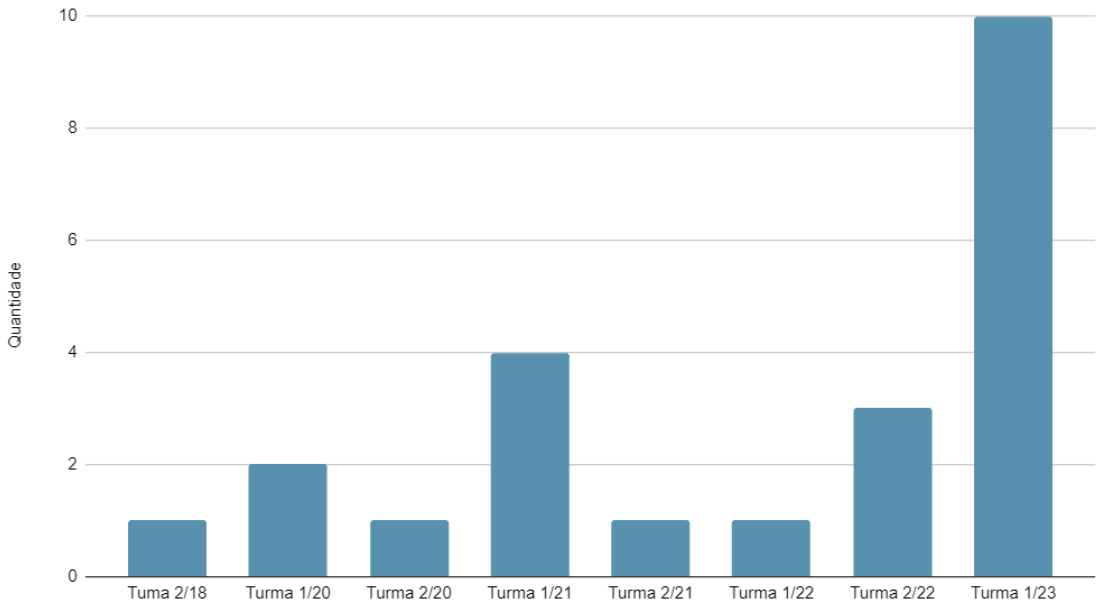


Figura 4.1: Gráfico de barras mostrando a relação de ano e semestre que cada aluno ingressou no curso

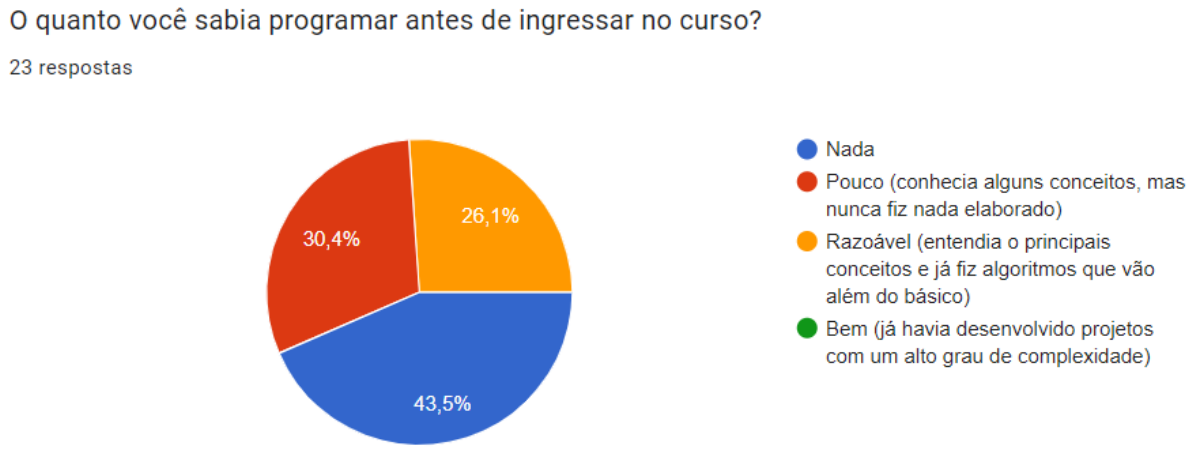


Figura 4.2: Gráfico de pizza com a separação de nível de conhecimento de programação indicado por cada aluno

Portanto, este dado pode indicar que uma parte considerável dos alunos que ingressam nos cursos de Computação não tem conhecimento de conceitos de programação.

- **Resposta questão 3**

  A terceira questão perguntava o nível de conhecimento de inglês de cada aluno antes de ingressar no curso. Se os alunos não possuíam grande conhecimento de programação ao ingressarem o curso, é possível observar pela Figura 4.3 o contrário a respeito do nível de conhecimento de inglês. Boa parte da turma afirmou ter um conhecimento médio ou bom da língua inglesa, que, como já foi explicado, é um importante componente para aprendizagem de programação. Ao todo, onze (47,8%) alunos disseram ter um alto nível de inglês e dez (43,5%) um nível médio, apenas um aluno(a) ter um baixo conhecimento, o mesmo para nenhum conhecimento. Uma ponderação sobre essa situação é discutida na Feção 4.5.

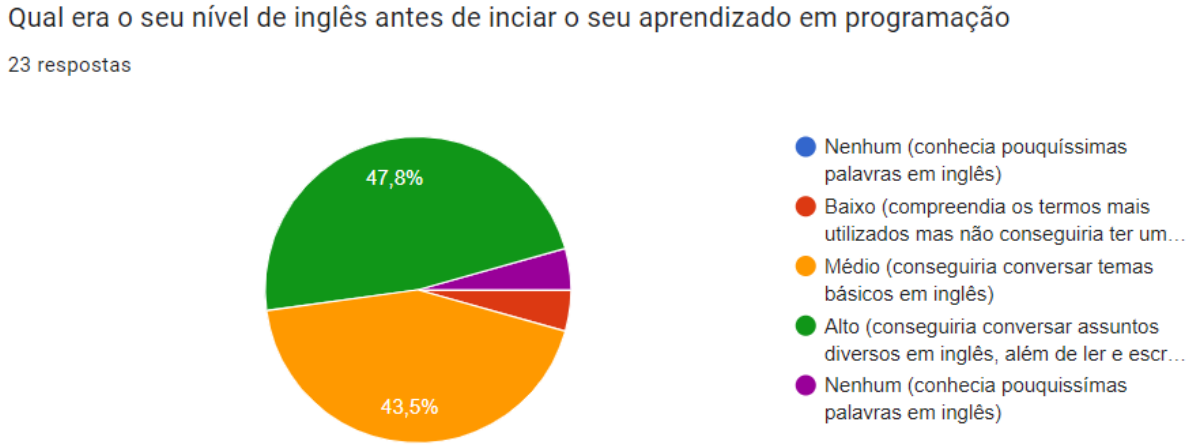


Figura 4.3: Gráfico de pizza com a separação de nível de conhecimento de de inglês ao entrar no curso de ciência da computação

- **Resposta questão 4**

  Na quarta questão a pergunta era sobre as principais dificuldades em APC. A partir desta questão, as repostas se dão de forma discursiva, portanto, para melhor representar todas as repostas será feito um agrupamento de respostas semelhantes e com isso um síntese geral de como determinada pergunta foi respondida.

  A questão quatro teve diversas reações diferentes, mas o tema mais presente foi ter problemas com "a lógica de programação". Isto pode indicar que os alunos conheciam os termos e instruções da linguagem com a qual estão trabalhando, mas não conseguiam traçar um raciocínio para escrever códigos assertivos, como relatado por um aluno(a): "eu sei como fazer mas não sei como mandar o computador fazer".

Sobre dificuldades de funcionalidades de linguagens de programação, as principais relatadas foram: recursividade, matrizes, condicionais, laço de repetição e noção matemática.

Foram relados também problemas de didática como o tempo para realizar provas e pouca motivação. Outros de alunos alegaram que "começar do zero" foi a sua maior dificuldade. Uma minoria relatou que tiveram apenas pequenos obstáculos ou até mesmo nenhuma dificuldade na disciplina introdutória do curso.

- **Resposta questão 5**

  Na questão de número 5 a pergunta foi direcionada a entender como foi a experiência com a utilização do Arara. Grande parte dos alunos relataram positivamente a sua interação com a aplicação Arara. É possível colocar dezoito alunos (78.26%) neste escopo.

  Dentre estes comentários, alguns acrescentaram críticas como uma estranheza inicial, a limitação da ferramenta, e eventuais bugs ocorridos. Por outro lado, foram adicionados elogios após a resposta, como ter sido fácil, intuitivo e divertido.

  Muitos alunos relataram uma estranheza por estarem acostumados apenas em linguagens de programação em inglês. Este é um ponto que poderia ser minimizado caso este teste na ferramenta Arara fosse aplicada para a disciplina introdutória do curso.

  Cinco alunos da turma relataram uma experiência ruim com a aplicação Arara, especialmente por se sentirem desconfortáveis e pouco motivados para trabalhar com uma linguagem em português.

- **Resposta questão 6**

  Na sexta questão foi perguntado se o software Arara apresentou problemas. Nesta pergunta foram submetidas vinte e duas repostas, uma a menos que a totalidade de alunos que responderam ao formulário.

  Dezessete alunos (73,9%) afirmaram que tiveram problemas ao interagir com o Arara, ao passo que cinco (21,7%) relatam não ter tido nenhum tipo de problema. Todos aqueles que afirmaram ter problemas e especificaram o que ocorreu, escreveram que os problemas estavam ligados essencialmente a linguagem e ao seu interpretador, não houve reclamações de outros componentes de aplicação, como: fluxo de login, acessar as páginas, bugs visuais, erros no processo de submissão de atividade, dentre outros.

Os principais problemas relatados estão relacionados com definição de escopos, utilização de parênteses, nomeação de variáveis e operações matemáticas. Esses podem ser entendidos da seguinte forma:

- Definição de escopos - Os alunos relataram que estruturas aninhadas (com um **se/senao/senao se** dentro de outra condicional) apresentavam problemas ou não funcionavam;
- Utilização de parênteses - Esta foi a reclamação mais apontada pelos alunos. Eles alegaram não conseguir utilizar parênteses para fazer operações aritméticas;
- Nomeação de variáveis - Problemas com a nomeação de variáveis e sua utilização também foi muito apontada pelo alunos. As principais queixas eram que ao criar variáveis com nomes genéricos como **a, b, re, etc.** apresentavam problemas ao executar o código ao passo que quando trocavam seus nomes para **resultado, saida, etc.** passava a funcionar;
- Operações Matemáticas - Os alunos alegram problemas especialmente na precedência de operadores.

Todos problemas relatados pelos alunos foram posteriormente testados e comprovados, ou seja, de fato estes problemas ocorrem e foram registrados como parte integrantes dos trabalhos futuros.

- **Resposta questão 7**

  A questão de número 7 perguntava a opinião do aluno sobre a utilidade do software Arara. Esta questão foi respondida por vinte e três alunos e contou com unanimidade de forma positiva. Todos os alunos consideraram que ela pode ser útil para a aprendizagem de programação. Mesmo assim alguns alunos fizeram ressalvas. As principais foram: "útil de forma limitada", "apenas no começo de aprendizagem mesmo", "ainda há muito o que melhorar" e "apenas para quem não sabe inglês".

- **Resposta questão 8**

  A oitava questão dizia respeito se a utilização do Arara teria ajudado no seu processo de aprendizagem de programação. Esta pergunta esperava colocar o aluno para dizer se ela seria útil para ele mesmo e não apenas para um terceiro. Onze alunos (47,8%) afirmaram que se tivessem utilizado o Arara no início do seu processo de aprendizagem teriam menos dificuldade no processo. Nove (39,1%) afirmaram que não teria colaborado no seu processo individual e três (13,1%) não souberam responder ou ficaram indecisos.

4.5 Análise dos resultados

Esta avaliação não foi pontuada na turma, até mesmo para não pressionar os alunos a responderem ao formulário e obter respostas mais sinceras. Porém, efeito indesejado disto é que na turma de quarenta e três alunos, apenas vinte e três alunos o responderam.

É importante destacar que os alunos que foram submetidos à este teste já são estudantes que aprenderam todos os conceitos básicos de programação há ao menos um semestre, visto que Estrutura de Dados é do segundo semestre do curso seguindo o fluxo. As respostas da primeira questão corroboram isto. Isto significa que é possível obter resultados diferentes caso esta pesquisa de opinião fosse realizada em alunos calouros.

Com base nas respostas dos alunos, talvez forneça um bom indicativo que os alunos ingressantes nos cursos de computação da UnB possuem um bom nível de inglês para a média brasileira, que é baixa como visto na Seção 1.1 [1], e não necessariamente sabem programar. Uma possível explicação para essa situação é que inglês é uma matéria obrigatória a partir do 6º ano escolar e programação não. Além disso, inglês geralmente é cobrado nas provas que dão acesso à universidades e, novamente, programação não.

Sobre as perguntas de usabilidade do Arara, foram relatados vários problemas acerca de seu interpretador. Isto já era esperado por ser um software grande e complexo de se construir em apenas um semestre. Os problemas de precedência de operadores e utilização de parênteses já estavam mapeados e era sabido que poderiam surgir problemas neste sentido. Por outro lado, todas as escolhas feitas sobre os temas como usabilidade e design se mostraram acertadas, uma vez que não houveram reclamações neste sentido.

Outro ponto interessante a ser observado é a grande diferença de respostas nas questões sete 4.4.2 e oito 4.4.2. Na primeira delas há um consenso na turma que a ferramenta pode sim ser útil na aprendizagem de programação (mesmo com ressalvas). Na segunda pergunta, a turma divide opiniões entre útil, não útil e incerteza. Há a possibilidade de tal situação ocorrer justamente por eles estarem em um contexto de dominarem a língua inglesa e alguns já saberem programar ao adentrar no curso.

[1] Loureço, Aline: Apenas 5% da população brasileira fala inglês, aponta pesquisa. https://www.segs.com.br/educacao/347834-apenas-5-da-populacao-brasileira-fala-ingles-aponta-pesquisa, acesso em 2023-09-07

Capítulo 5

Considerações Finais

Durante o desenvolvimento deste trabalho foi realizado o desenvolvimento de uma ferramenta com bastante potencial de ajudar muitas pessoas em seus primeiros passos no mundo da programação. Esta ferramenta foi inicialmente pensada para auxiliar os ingressantes em cursos de computação da Universidade de Brasília (UnB). Entretanto, com indicativos e feedbacks de alunos e professores talvez ela poderá ser adaptada para o focar em momentos mais iniciais da aprendizagem de programação.

5.1 Limitações

Como levantado na Seção 3.4.3, algumas limitações já estavam mapeadas. Porém, como visto no estudo piloto, há algumas limitações a respeito do interpretador Arara, isto gera uma quantidade significativa de problemas. A ferramenta Arara ainda carece de ajustes e não está totalmente pronta para ser utilizada em larga escala. Outra limitação que poder ser levantada foi ter feito apenas um estudo, o ideal seria a realização de mais estudos com diversos outros grupos.

5.2 Lições aprendidas

Com a experiência adquirida com a realização deste projeto e a aplicação do estudo piloto é possível destacar três principais lições aprendidas.

- Posicionamento do Arara - Como foi visto na Seção 4.5, os alunos de computação da UnB que participaram do estudo piloto do Arara possuem um alto nível de inglês, em média, isto significa que pode-se tornar menos interessante a utilização da aplicação Arara por um longo período. Talvez o mais adequado ponto para se utilizar este projeto é apenas na fase introdutória da disciplina onde são explicados

os conceitos mais básicos de programação e com a aplicação sistemática de tarefas pelo professor. Desta maneira pode-se extrair mais da ferramenta no contexto de uma universidade federal brasileira. Mas é possível que a ferramenta se sai melhor em ambientes onde o domínio de inglês é menor, como no ensino fundamental e médio, especialmente público, e em faculdades particulares com um pior ranking, por exemplo.

- Necessidade de melhorias - Levando em consideração tudo aquilo que foi observado no estudo piloto, a ferramenta tem potencial de ser útil, mas necessita de correções em seu interpretador.
- Manutenção do visual - A respeito de usabilidade e escolha visual não houveram quaisquer críticas, apenas alguns elogios. Portanto, esta parte deve ser mantida como um todo e apenas ajustes finos ou adições devem ser feitos.
- Uso de testes unitários - Durante o desenvolvimento não foram utilizados testes unitários. A utilização deste tipo de técnica geraria uma grande ganho de produtividade e reduziria o retrabalho ao corrigir um ponto do software e danificando outro.

5.3 Trabalhos futuros

O Arara é um software que ainda está em um estágio inicial e que necessita de ajustes e melhorias. Baseado em tudo que foi analisado até este ponto, existem dois principais pontos para se trabalhar neste projeto.

O primeiro deles e o mais importante é o de corrigir todas os problemas relatados pelos aluno durante a sua utilização. Ainda complementar com a adição de novas funcionalidades ao interpretador conforme os desenvolvedores julgarem necessário.

O segundo ponto é o de aplicar novos estudos pilotos já com correções feitas em diferentes tipo de grupos de pessoas, como: turmas de de ensino fundamental e médio, de escolas públicas e particulares, turmas de introdução do Departamento de Ciência da Computação (CIC), turmas de faculdades particulares. Isto traria informações relevantes do comportamento do Arara com diferentes populações.

Após isso, deve-se colher feedbacks, analisá-los e aplicar aquilo que for necessário e agregador ao software Arara. Para que com isto ele possa ajudar alunos com dificuldades iniciais e promover um ambiente mais favorável para estudantes de computação.

Com todo este processo realizado, caso os estudos direcionem que seria mais proveitoso utilizar o software Arara com foco em um destes grupos, fazer adaptações direcionadas às

necessidades deste grupo escolhido. Além de saber o nível de inglês, também seria interessante pesquisar o quanto o nível de inglês interfere na aprendizagem de programação.

Outro ponto interessante seria o de mensurar o nível de inglês de alunos brasileiros ingressantes nos cursos de computação de universidades públicas e privadas. Isto poderia dar informações valiosas que poderiam direcionar melhor o desenvolvimento do Arara. Seguindo nessa linha, seria interessante também estudar também se a aprendizagem de programação utilizando linguagem em inglês em alunos que dominam esta língua se equivale ao utilizar a sua língua materna.

Referências

[1] Aho, Alfred V., Ravi Sethi e Jeffrey D. Ullman: *Compilers: Principles, Techniques and Tools*. Addison Wesley Publishing Company, 2006. x, 9, 10, 11, 12, 13, 14

[2] Bertolini, Cristiano, Fábio José Parreira, Guilherme Bernardino da Cunha e Ricardo Tombesi Macedo: *Linguagem de programação i*. 2019. xi, 9, 10, 16, 17, 18, 20

[3] Kumar, Ashok: *Teaching english based programming courses to english learners/non-native speakers of english*. fevereiro 2014. 2

[4] Valéria, Márcia e César França: *Ferramentas de auxílio ao aprendizado de programação : Um estudo comparativo*. janeiro 2013. 2

[5] Beaubouef, Theresa e John Mason: *Why the high attrition rate for computer science students: Some thoughts and observations*. SIGCSE Bulletin, 37:103–106, junho 2005. 3

[6] Maceda, Gabriela Sivlva, Pedro Arjona Villicana e Francisco Edgar Castillo Barrera: *More time or better tools? a large-scale retrospective comparison of pedagogical approaches to teach programming*. IEEE Transactions on Education, 59:1–8, março 2016. 3

[7] Canedo, Edna Dias, Giovanni Almeida Santos e Sérgio Antonio Andrade de Freitas: *Analysis of the teaching-learning methodology adopted in the introduction to computer science classes*. Em *2017 IEEE Frontiers in Education Conference (FIE)*, páginas 1–8, 2017. 3

[8] Bez, Jean Luca, Neilor A. Tonin e Paulo R. Rodegheri: *Uri online judge academic: A tool for algorithms and programming classes*. Em *2014 9th International Conference on Computer Science & Education*, páginas 149–152, 2014. 4

[9] Manso, Antonio, Luís Lopes, Célio Marques, Raquel Guedes e Paulo Santos: *Algorithmi ide -integrated learning environment for the teaching and learning of algorithmics*. janeiro 2020. 4

[10] Yang, Jeong, Young Lee, David Hicks e Kai Chang: *Enhancing object-oriented programming education using static and dynamic visualization*. páginas 1–5, outubro 2015. 4

[11] Ramos, Geovana: *Calango*. `https://github.com/GeovanaRamos/calango`, acesso em 2023-07-17. 6, 19, 23, 26, 27

[12] Indrusiak, Leandro Soares: *Linguagem java*. Grupo JavaRS JUG Rio Grande do Sul, página 19, 1996. 15

[13] McConnell, Steve: *Code Complete, Second Edition*. Microsoft Press, USA, 2004, ISBN 0735619670. 15

[14] Kurnia, Andy, Andrew Lim e Brenda Cheang: *Online judge*. Computers & Education, 36(4):299–315, 2001. 16, 17

[15] Revilla, Miguel A, Shahriar Manzoor e Rujia Liu: *Competitive learning in informatics: The uva online judge experience*. Olympiads in Informatics, 2(10):131–148, 2008. 16

[16] Bagno, Marcos: *Linguagem*. Glossario Ceale, 2014. 17

[17] Flanagan, David: *JavaScript: o guia definitivo*. Bookman Editora, 2012. 17, 18

[18] Van Roy, Peter *et al.*: *Programming paradigms for dummies: What every programmer should know*. New computational paradigms for computer music, 104:616–621, 2009. 18, 19

[19] Ricarte, Ivan Luiz Marques: *Programação orientada a objetos: uma abordagem com java*. http://www. dca. fee. unicamp. br/cursos/PooJava/Aulas/poojava. pdf> Acesso em, 29(10):2014, 2001. 19

[20] Wadler, Philip: *The essence of functional programming*. Em *Proceedings of the 19th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, páginas 1–14, 1992. 19

[21] Silva, Dilma Menezes da: *Introdução à programação concorrente para a internet*. 1999. 19

[22] Gadêlha, Douglas: *Portugol Webstudio*. `https://dgadelha.github.io/Portugol-Webstudio`, acesso em 2023-06-18. 20, 23, 27, 31

[23] Gomesh, David e Elias Lima: *Mapler*. `https://portugol.sourceforge.io`, acesso em 2023-09-17. 20, 27

[24] Santos Soares, Antônio Vinícius Menezes Medeiros e Prof. Dr. Michel dos: *Portugol Online*. `https://vinyanalista.github.io/portugol`, acesso em 2023-09-17. 20, 27

[25] Wainer, Jacques e Eduardo C Xavier: *A controlled experiment on python vs c for an introductory programming course: Students' outcomes*. ACM Transactions on Computing Education (TOCE), 18(3):1–16, 2018. 21

[26] Tanenbaum, Andrew S. e Herbert Bos: *Modern Operating Systems*. Pearson, Boston, MA, 4ª edição, 2014, ISBN 978-0-13-359162-0. 23

[27] Murillo, Danny: *Navegadores web*. El Tecnológico, 27(1):13–14, 2017. 23

[28] Manger, Christian, Tomasz Trejderowski e Jarosław Paduch: *Advantages and disadvantages of framework programming with reference to yii php framework, gideon. net framework and other modern frameworks*. Studia Informatica, 31(4A):119–137, 2010. 24

[29] Rawat, Prateek e Archana N Mahajan: *Reactjs: A modern web development framework*. International Journal of Innovative Science and Research Technology, 5(11):698–702, 2020. 24

[30] Brace, Ian e Kate Bolton: *Questionnaire Design: How to Plan, Structure and Write Survey Material for Effective Market Research*. KoganPage, London ; New York, NY, fifth edition edição, 2022, ISBN 978-1-398-60414-8 978-1-398-60412-4. 45

Apêndice A

Análise e códigos do estudo de comparação das linguagens de programação

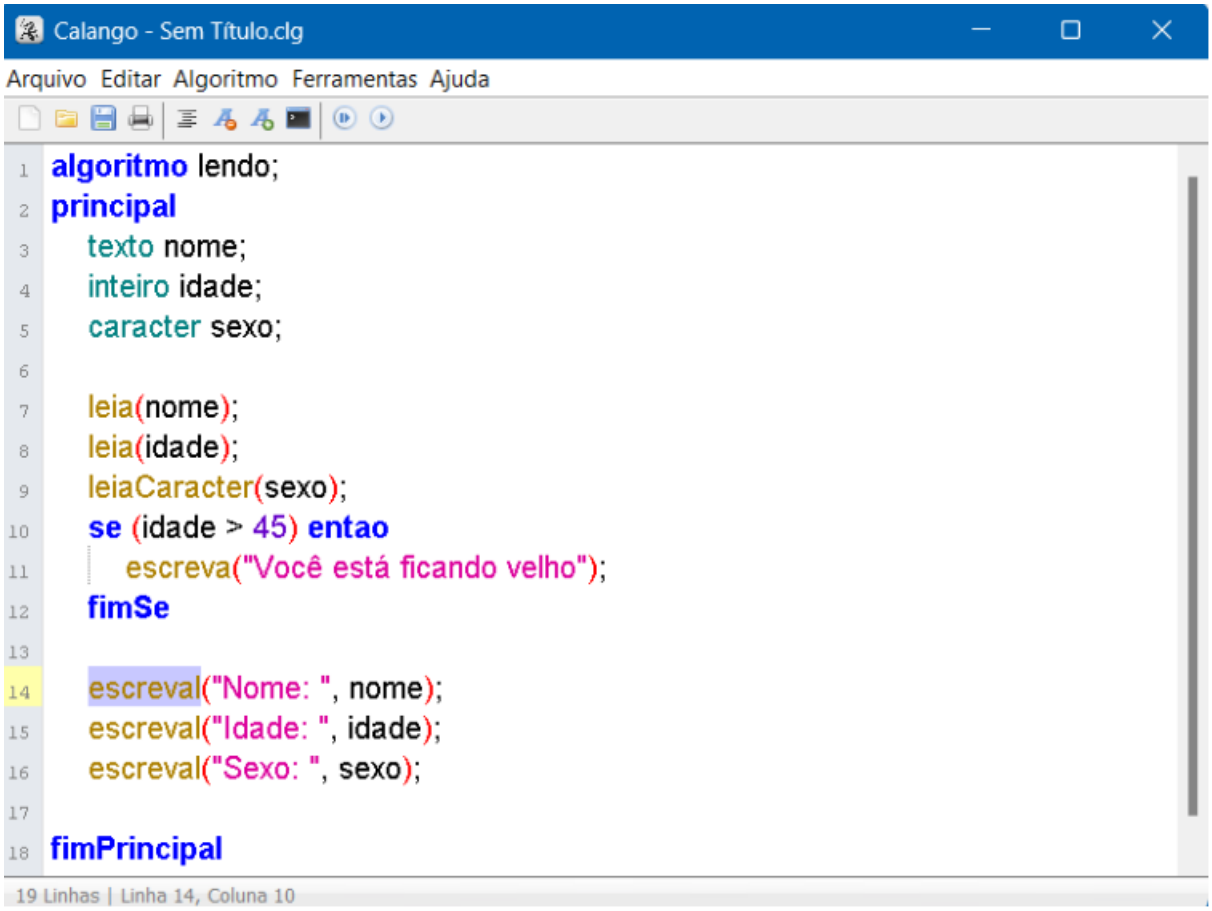
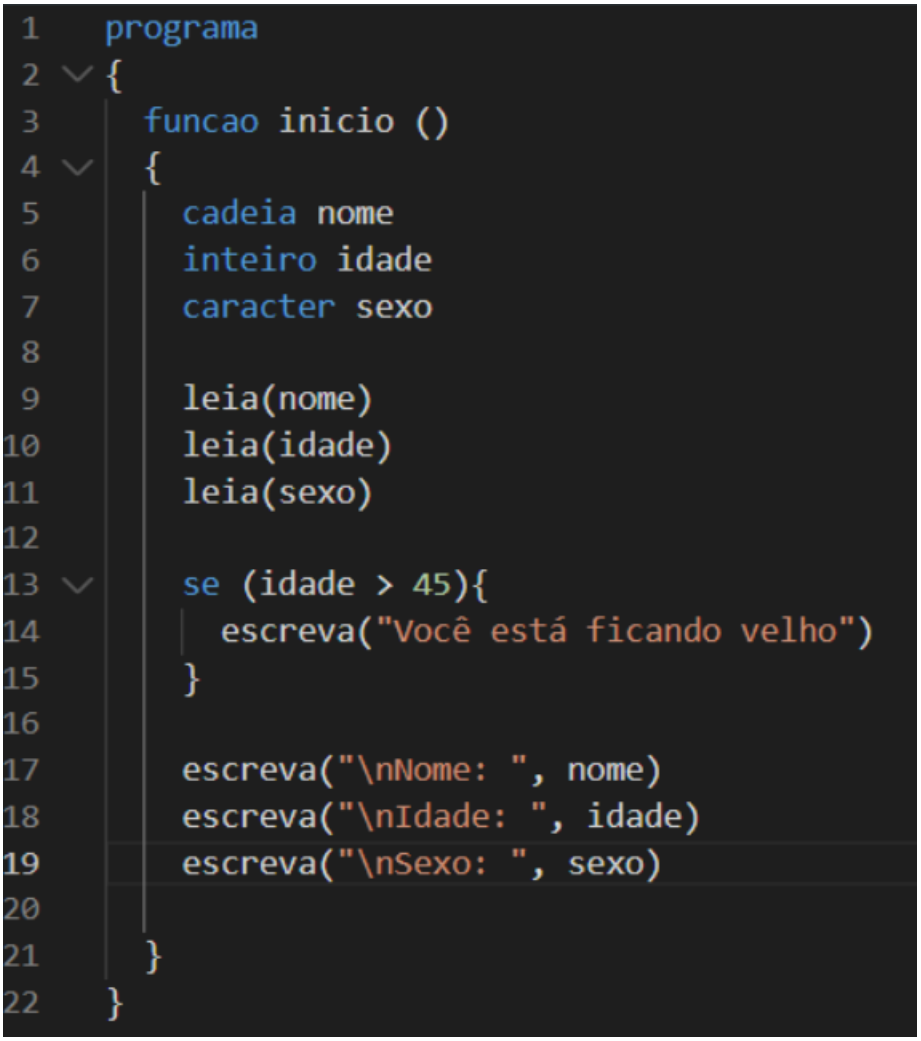


Figura A.1: Código exemplo com a linguagem Calango



```
programa
{
  funcao inicio ()
  {
    cadeia nome
    inteiro idade
    caracter sexo

    leia(nome)
    leia(idade)
    leia(sexo)

    se (idade > 45){
      escreva("Você está ficando velho")
    }

    escreva("\nNome: ", nome)
    escreva("\nIdade: ", idade)
    escreva("\nSexo: ", sexo)

  }
}
```

Figura A.2: Código exemplo com a linguagem Portugol WebStudio

```c
#include <stdio.h>

int main() {
    char nome[50];
    int idade;
    char sexo;

    printf("Digite seu nome: ");
    scanf("%s", nome);
    printf("Digite sua idade: ");
    scanf("%d", &idade);
    printf("Digite seu sexo: ");
    scanf(" %c", &sexo);

    if(idade > 45){
        printf("Você está ficando velho\n");
    }

    printf("Nome: %s\n", nome);
    printf("Idade: %d\n", idade);
    printf("Sexo: %c\n", sexo);
}
```

Figura A.3: Código exemplo com a linguagem C

```python
nome = input('Digite seu nome: ')
idade = input('Digite sua idade: ')
sexo = input('Digite seu sexo: ')

if idade > 45:
    print('Você está ficando velho!')

print('Nome: ', nome)
print('Idade: ', idade)
print('Sexo: ', sexo)
```

Figura A.4: Código exemplo com a linguagem Python

```kotlin
fun main() {

    println("Digite seu nome: ")
    val nome = readlnOrNull()

    println("Digite sua idade: ")
    val idade = Integer.valueOf(readlnOrNull());

    println("Digite seu sexo: ")
    val sexo = readlnOrNull()

    if (idade > 45){
        println("Você está ficando velho!")
    }

    println("Nome: $nome")
    println("Idade: $idade")
    println("Sexo: $sexo")

}
```

Figura A.5: Código exemplo com a linguagem Kotlin

Tabela A.1: Análise comparativa com base no estudo realizado na seção 2.6

	Calango	Portugol WebStudio	C	Python	Kotlin
Verbosidade	Pouco	Médio	Médio/Alto	Muito pouco	pouco
Tipo de atribuição	Tem que declarar sem valor. Tem que declarar antes de usar	Pode declarar sem valor. Tem que declarar antes de usar	Pode declarar sem valor. Pode declarar antes de usar.	Pode declarar sem valor. Não que declarar antes de usar.	Pode declarar sem valor. Não que declarar antes de usar.
Tipagem	Forte	Forte	Forte	Fraca	Forte
Bibliotecas Frameworks	Não possui	Possui algumas bem simples do próprios criador	Diversas	Diversas	Diversas
Método de Tradução	Interpretado	Interpretado	Compilado	Interpretado	Interpretado
Complexidade de Aprendizagem	Fácil/Médio	Fácil/Médio	Difícil	Muito fácil	Médio
Definição de escopo	Token - Fim<Token>ves	Com chaves	Com chave	Com indentação	Com chaves

Apêndice B

Respostas ao formulário

Tabela B.1: Tabela de respostas dos alunos no formulário do estudo piloto. Parte 1 de 4

Alunos	**Em que ano e semestre você ingressou no curso?**	**O quanto você sabia programar antes de ingressar no curso?**	**Qual era o seu nível de inglês antes de inciar o seu aprendizado em programação?**
aluno 1	2023.1	Nada	Médio (conseguiria conversar temas básicos em inglês)
aluno 2	2023.1	Nada	"Alto (conseguiria conversar assuntos diversos em inglês, além de ler e escrever textos)"
aluno 3	2023 no primeiro semestre	Nada	"Alto (conseguiria conversar assuntos diversos em inglês, além de ler e escrever textos)"
aluno 4	2023/1	Nada	"Alto (conseguiria conversar assuntos diversos em inglês, além de ler e escrever textos)"
aluno 5	"2022,1"	Nada	Médio (conseguiria conversar temas básicos em inglês)
aluno 6	2022/2	Razoável (entendia o principais conceitos e já fiz algoritmos que vão além do básico)	"Alto (conseguiria conversar assuntos diversos em inglês, além de ler e escrever textos)"
aluno 7	2023/1	Razoável (entendia o principais conceitos e já fiz algoritmos que vão além do básico)	Médio (conseguiria conversar temas básicos em inglês)
aluno 8	2018/2	"Pouco (conhecia alguns conceitos, mas nunca fiz nada elaborado)"	Médio (conseguiria conversar temas básicos em inglês)
aluno 9	2020/01	Nada	Baixo (compreendia os termos mais utilizados mas não conseguiria ter uma conversa simples em inglês)
aluno 10	2023 - 1 semestre	Nada	Médio (conseguiria conversar temas básicos em inglês)
aluno 11	2022/2	Nada	"Alto (conseguiria conversar assuntos diversos em inglês, além de ler e escrever textos)"
aluno 12	2023/1	"Pouco (conhecia alguns conceitos, mas nunca fiz nada elaborado)"	Nenhum (conhecia pouquissímas palavras em inglês)
aluno 13	1/2020	Razoável (entendia o principais conceitos e já fiz algoritmos que vão além do básico)	Médio (conseguiria conversar temas básicos em inglês)
aluno 14	2023-1	"Pouco (conhecia alguns conceitos, mas nunca fiz nada elaborado)"	"Alto (conseguiria conversar assuntos diversos em inglês, além de ler e escrever textos)"
aluno 15	2021/1	"Pouco (conhecia alguns conceitos, mas nunca fiz nada elaborado)"	"Alto (conseguiria conversar assuntos diversos em inglês, além de ler e escrever textos)"
aluno 16	2021.2	Nada	Médio (conseguiria conversar temas básicos em inglês)
aluno 17	01/2023	Razoável (entendia o principais conceitos e já fiz algoritmos que vão além do básico)	"Alto (conseguiria conversar assuntos diversos em inglês, além de ler e escrever textos)"
aluno 18	2021/01	Razoável (entendia o principais conceitos e já fiz algoritmos que vão além do básico)	"Alto (conseguiria conversar assuntos diversos em inglês, além de ler e escrever textos)"
aluno 19	22.1	Razoável (entendia o principais conceitos e já fiz algoritmos que vão além do básico)	"Alto (conseguiria conversar assuntos diversos em inglês, além de ler e escrever textos)"
aluno 20	2022.2	"Pouco (conhecia alguns conceitos, mas nunca fiz nada elaborado)"	Médio (conseguiria conversar temas básicos em inglês)
aluno 21	2020/2	"Pouco (conhecia alguns conceitos, mas nunca fiz nada elaborado)"	Médio (conseguiria conversar temas básicos em inglês)
aluno 22	2023.1	"Pouco (conhecia alguns conceitos, mas nunca fiz nada elaborado)"	"Alto (conseguiria conversar assuntos diversos em inglês, além de ler e escrever textos)"
aluno 23	2023/1	Nada	Médio (conseguiria conversar temas básicos em inglês)

Tabela B.2: Tabela de respostas dos alunos no formulário do estudo piloto. Parte 2 de 4

Alunos	**Qual foi a sua principal dificuldade na disciplina de Algoritmos e Programação de Computadores - APC?**	**Como a foi sua experiência com o software Arara?**
aluno 1	A minha maior dificuldade foi a lógica de programação e saber articular isso.	"Achei super interessante, uma experiência única, estressa um pouquinho por ser algo diferente , mas adorei. Talvez por a gente tá acostumado com Python foi estranho kkkkkjj, pq toda escrevia na formatação em python por ser algo automático. Mas em geral, é uma ferramenta bem dinâmica e interessante de se utilizar."
aluno 2	Começar a aprender do zero	"Estranha, pois eu estava acostumado com os termos em inglês (if, else, print, input, etc)"
aluno 3	"A minha principal dificuldade foi no início quando eu não sabia como programar nada, então o início eu tive que esforçar mais para aprender e entender."	"O software é bom mas apresenta algumas limitações, pois não compreende parênteses nos cálculos matemáticos e também a falta de uma ferramenta de debug dificulta um pouco a correção dos códigos no software."
aluno 4	"Pensar em algoritmos capazes de resolver os problemas, e um pouco da linguagem. Ainda não sei como aplicar o conhecimento da disciplina"	"Bom ler a documentação (sucinta)
aluno 5	Programar com a pressão de tempo em uma prova	Boa
aluno 6	a parte de dicionário	foi uma boa experiencia
aluno 7	"Pensamento computacional para resolver os problemas	o famoso eu sei como fazer mas não sei como mandar o computador fazer."
aluno 8	Disciplina e dedicação	"A primeira vista, tranquilo, fácil de entender, porém limitado."
aluno 9	"Para ser sincero, não me lembro. Pois me identifiquei de mais (gostei muito), se for ver dificuldade, posso dizer um tema: recursão, mas não uma dificuldade."	"Legal, pois vem com uma abordagem diferente, porém chata por conta da sintaxe, mas um chato normal, algo comum de se conhecer uma nova linguagem. Justamente por já ter certa experiência em outra linguagem (fator extremamente importante)."
aluno 10	"lista, dicionário e Matriz"	legal
aluno 11	"Comprometimento?, no semestre que entrei as listas não valiam ponto o que era desestimulante para faze-las, a professora que tive também não ajudou."	"Tranquila, foi minha primeira vez programando fora de python mas não senti dificuldades."
aluno 12	Programar	Boa
aluno 13	Recursão	"Boa, apesar do bugs de não poder usar parênteses no código que criei."
aluno 14	Debugar códigos	Boa
aluno 15	matrizes	"sou mais acostumado com python, então a similaridade com C me atrapalhou "
aluno 16	Compreender como funcionava os laços.	"Gostei por usar a língua portuguesa, não demorei pra entender como funcionava por esse fato."
aluno 17	Lidar com prazos de entrega de trabalhos e atividades	Razoavelmente boa
aluno 18		Achei interessante porém limitado
aluno 19	Conseguir fazer mais questões da prova	Foi uma experiência um pouco caótica porém interessante
aluno 20	Matrizes e noções matemáticas	Ótima oportunidade para a introdução a aprendizagem da programação.
aluno 21	recursividade if-else	Nem um pouco acostumado com uma linguagem em pt-br
aluno 22	Encontrar erros mais fundamentais do código que o faziam falhar casos de teste escondidos.	"Inicialmente ruim, pois estou acostumado com outras linguagens de programação. "
aluno 23	A lógica e a semântica.	Achei uma ideia muito boa e a interface é bem intuitiva. A sintaxe e a semântica permitem que o código seja criado de forma mais rápida.

Tabela B.3: Tabela de respostas dos alunos no formulário do estudo piloto. Parte 3 de 4

Alunos	"O software Arara apresentou problemas? Caso positivo, quais?"	Acredita que a ferramenta pode ser útil para aprendizagem de programação?
aluno 1	"Só que teve algumas coisas que precisam ser vistas, tipo a questão da variável, lá tinha explicando, só que na hora de armazenar uma variável printar ela dava erro, pois ela não printava o valor que estava na variável, mas sim o nome. Tem coisas também na questão do se e senao que estava dando alguns erros, que eu n soube identificar e a estrutura estava igual a da documentação."	"Sim, acredito muito. É uma forma interessante de aprender, porque te faz pensar bastante, já que é algo mais compactado."
aluno 2	"Sim, não conseguia fazer um if dentro de um if e não conseguia fazer um loop dentro do if e vice e versa"	"Sim, mas há muito o que melhorar, por exemplo, não dá pra implementar funções"
aluno 3	"Apresentou alguns problemas ao nomear as variáveis, pois às vezes o software não compreendia certas letras como nomes de variáveis."	"Sim, acredito que seja capaz de ajudar bastante na aprendizagem por ser um software simples."
aluno 4	"Sim. No problema de transformar uma temperatura em graus fahreinheint para graus celsius, o programa não funcionou com a variável dos graus celsius nomeada como “c”. Quando mudei pra “resultado”, no mesmo código, funcionou "	"Sim, principalmente para nível de ensino básico. Linguagem muito simples e didática "
aluno 5	"sim, muito bug em {} e ()"	Sim
aluno 6	não	"com certeza, acredito que a ferramenta possa servir para pessoas que querem aprender sua primeira linguagem que não possuam um vasto conhecimento em inglês."
aluno 7	não apresentou nenhum problema quando fui utilizá-lo.	"com um guia mais geral de como usar acredito que possa sim ser útil para aprendizagem, mas atualmente é necessário um professor ou monitor para te guiar nos passos iniciais do aplicativo."
aluno 8	"Sim. O ponto principal seria a impossibilidade de mais de uma operação por cada par de objetos, o que limita bastante e torna cansativo a programação."	"Sim, ainda que de forma limitada."
aluno 9	"Talvez falte mais informações na documentação Arara	como se aninhados
aluno 10	Não	"Sim, muito, da mesma forma como também é útil ensinar uma base de lógica no visualg"
aluno 11	Não	Sim
aluno 12	Não apresentou	Sim
aluno 13	"O bug do cálculo, de não conseguir ler os parênteses: 5 * ((F-32) / 9)"	Sim.
aluno 14	Se eu não me engano em operações matemáticas	Sim
aluno 15	não	sim pra quem não sabe inglês
aluno 16	"Problemas não, mas coisas a melhorar pro melhor entendimento sim, como por exemplo o não uso de parênteses ou algo que simbolize nas expressões ficou meio confuso."	"Sim, a proposta é muito interessante para pessoas que querem aprender a programar do zero"
aluno 17	"Sim, houve problemas no uso de parenteses e também a falta de alguns recursos."	"Certamente, o software Arara me pareceu muito com o visualg, mas com uma interface gráfica mais amigável, talvez com algumas melhoras na documentação e adição de recursos ela seja completamente funcional como ferramenta de ensino."
aluno 18	"Sim, execução de apenas uma operação por linha"	Acredito que sim para pessoas que chegam completamente sem conhecimentos mas eu particularmente prefiro começar em C ou python do que algo como portugol
aluno 19	"Algumas variáveis não podem ter certos nomes	como a ou e; Não é possível fazer mais de uma operação matemática em uma linha de código; A linguagem não reconhece parentes ou chaves."
aluno 20	"Alguns, inclusive reportei diretamente no suporte."	"Sim, com certeza. Através de uma lógica didática ela será de uso indispensável em sala de aula."
aluno 21	Não me recordo	"Sim, principalmente para quem não domina inglês ou para crianças terem o contato"
aluno 22		"Acredito, especialmente para aqueles que não possuem proficiência no inglês ."
aluno 23	"As condicionais não estavam funcionando corretamente. O senão mesmo colocado após o se não funcionava	apenas houve uma mensagem de erro informando que não havia um se para o senão."

Tabela B.4: Tabela de respostas dos alunos no formulário do estudo piloto. Parte 4 de 4

Alunos	Acredita que se tivesse começado a aprender programação utilizando o software Arara teria tido menos dificuldade no caminho?
aluno 1	"No meu caso acredito que sim, pq como é algo feito na nossa linguagem entenderíamos melhor, mas depende muito também, porque foi bem difícil no início ali, porém com a prática ficaria mais tranquilo com o tempo."
aluno 2	"Não tenho certeza, pois os videos do Guanabara me ajudaram bastante pra uma pessoa que não sabia nada. Entretanto, achei bem detalhado a parte da documentação nos comandos apresentados e de fácil compreensão, além da plataforma ser bonita. "
aluno 3	"Sim, talvez eu tivesse menos dificuldade justamente pelo software ter uma linguagem simplificada."
aluno 4	Não. Acho que Python é simples o suficiente para introdução da programação
aluno 5	Não
aluno 6	acredito que eu teria menos dificuldade de passar para outras linguagens como C por conta da minha primeira linguagem foi o próprio python e acredito que isso me deixou um pouco mal acostumado pois o python tem diversas facilidades como não precisar falar qual será a variavel recebida e não precisar de ; alem de diversas outras facilidades que em outras linguagens é mais chatinho.
aluno 7	"Talvez, não sei dizer."
aluno 8	"Não. A ferramenta ainda está muito atrás de outras ferramentas que estão disponíveis em outro locais. Para um projeto é algo que possa se desenvolver, mas não chega ao nível ainda que poderia se tornar algo que facilitasse o aprendizado. Uma alternativa seria tornar comandos mais abrangentes, instruções mais claras para o ensino, além de algo que possa instigar a sede para o aprendizado dentro da plataforma."
aluno 9	Com certeza.
aluno 10	provavelmente sim
aluno 11	"Acredito que sim, pois provê uma aproximação mais 'humana' a programação."
aluno 12	Sim
aluno 13	"Acredito que não, não tive muita dificuldade no começo."
aluno 14	"Provavelmente
"	
aluno 15	não
aluno 16	"Sim, acho que o software é uma boa proposta para ser implementado nas escolas como já uma forma de preparação caso o aluno escolha tal área."
aluno 17	"Com certeza, como é uma linguagem que se aproxima da nossa língua materna tornaria a compreensão sobre programação mais clara."
aluno 18	Acho que não
aluno 19	Creio que não
aluno 20	Com certeza.
aluno 21	"Sim, pois a documentação é bem intuitiva. Dá para copiar os exemplos e compilar o código sem nenhuma dificuldade!"
aluno 22	"Não, dado que possuo proficiência no inglês."
aluno 23	"Acredito que haveria um aproveitamento melhor do que usar um simples portugol, além de ambientar melhor os alunos na programação."