UNIVERSIDADE FEDERAL DO MARANHÃO
DEINF - DEPARTAMENTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

LEONARDO WILLIAM MACHADO SILVA

MACP UMA EXPERIÊNCIA COM SOFTWARE LIVRE NA MELHORIA DE UMA FERRAMENTA PARA PORTUGUÊS ESTRUTURADO

SÃO LUÍS
2017

LEONARDO WILLIAM MACHADO SILVA

MACP UMA EXPERIÊNCIA COM SOFTWARE LIVRE NA MELHORIA DE UMA FERRAMENTA PARA PORTUGUÊS ESTRUTURADO

apresentado ao Curso de Ciência da Computação da Universidade Federal do Maranhão, como requisito parcial para a obtenção do título de Bacharel.

Orientador: Carlos de Salles Soares Neto
Universidade Federal do Maranhão

SÃO LUÍS
2017

SILVA, LEONARDO WILLIAM MACHADO.
MACP UMA EXPERIENCIA COM SOFTWARE LIVRE NA MELHORIA DE UMA FERRAMENTA PARA PORTUGUES ESTRUTURADO / LEONARDO WILLIAM MACHADO SILVA. – 2017.
34 f.

Orientador(a): CARLOS DE SALLES SOARES NETO.
Monografia (Graduação) – Curso de Ciência da Computação, Universidade Federal do Maranhão, SÃO LUÍS, 2017.

1. PORTUGOL. 2. PORTUGUÊS ESTRUTURADO. 3. PSEUDOCÓDIGO. 4. PSEUDO-LINGUAGEM. I. NETO, CARLOS DE SALLES SOARES. II. Título.

Leonardo William Machado Silva

MACP UMA EXPERIÊNCIA COM SOFTWARE LIVRE NA MELHORIA DE UMA FERRAMENTA PARA PORTUGUÊS ESTRUTURADO

Monografia apresentada ao Curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Trabalho aprovado. São Luís, 18 de JULHO de 2017.

Prof. Dr. Carlos de Salles Soares Neto (Orientador)
Doutor em Ciência da Computação
Universidade Federal do Maranhão

Prof. Dr. Mário Antonio Meireles Teixeira
Doutor em Ciência da Computação e Matemática Computacional
Universidade Federal do Maranhão

Prof. Dr. Tiago Bonini Borchartt
Doutor em Ciência da Computação
Universidade Federal do Maranhão

Prof. Me. Daniel Lima Gomes Júnior
Mestre em Engenharia de Eletricidade
Universidade Federal do Maranhão

São Luís
2017

Aos meus familiares e amigos.

AGRADECIMENTOS

Chega ao fim mais uma etapa do contínuo processo de busca do conhecimento e aprendizado. Nesse momento é que geralmente se para e faz - se então toda uma reflexão que apesar de todos os percalços encontrados na caminhada vê que tudo valeu apena. Observa - se então na saída que muitas dúvidas foram sanadas, muitas permanecem e novas são incorporadas, vemos também que em algum momento da caminhada mudamos e não somos mais as mesmas pessoas que éramos no início da jornada.

É a hora de reconhecer e agradecer a todos que fizeram parte dessa jornada, que de forma direta ou indireta contribuíram para que esse momento acontecesse lembrar que a vida são momentos sejam bons ou ruins e que de alguma forma eles contribuíram no processo.

Agradeço aos professores que em seu esforço procuram sempre informar e preparar, pois não transmitem apenas conteúdo acadêmico mais opiniões e histórias de vida não esquecendo os colegas que partilharam vários momentos durante a caminhada, observando que alguns vão além das fronteiras da instituição e das relações de simples coleguismo transformado se em grandes amigos.

Nesse momento reflexivo é que vemos o quão importante são os familiares, pois eles são quem geralmente nos apoiam quando estamos em dificuldades exaltando a capacidade que temos e que após tantos passos dados estamos mais próximos do objetivo pretendido do que do ponto em que iniciamos.

Agradeço em especial minha avó, Irene Maria Machado, a qual sempre ensinou a ser uma pessoa de caráter, a minha mãe, Meirilene Francisca Machado, que sempre me incentivou nos meus objetivos e a Claudiny Priscila Lopes Brito, que me auxiliou na conclusão dessa etapa.

"A dúvida é o principio da sabedoria" (Aristóteles)

"Uma longa viagem começa com um único passo." (Lao-Tsé)

RESUMO

SILVA, Leonardo. MACP uma Experiência com Software Livre na Melhoria de uma Ferramenta para Português Estruturado. 2017. 34 f. – Curso de Ciência da Computação, Universidade Federal do Maranhão. São Luís, 2017.

Uma das grandes dificuldades para os iniciantes em programação esta na dificuldade de assimilar os conceitos básicos relacionados à resolução de problemas utilizando a descrição algorítmica em português (pseudo-códigos). Ao utilizar uma linguagem simples como o Portugol, surge a necessidade da criação de uma ferramenta que possa executar os algoritmos criados a partir desse formalismo tornando o processo de aprendizado mais prático.
É muito importante ter disponível uma ferramenta confiável, que atenda as necessidades dos usuários. Muitos desenvolvedores tornam públicos os códigos fontes de desenvolvimento. Com isso pessoas interessadas com o projeto podem, de alguma forma, contribuir, seja através de correções de problemas identificados ou por inserções de funcionalidades, que possam ser úteis no processo de aprendizado.
A proposta de uma abordagem de desenvolvimento compartilhado ou livre faz com que vários programadores estejam contribuindo, aumentando em tese a velocidade ao qual a ferramenta é desenvolvida onde uma quantidade maior de olhos voltados para o projeto possa minimizar as taxas de erros existentes na versão disponível para os usuários.
O MACP hoje tem seus esforços de desenvolvimento baseados nessa política de código livre para que programadores interessados com o projeto possam contribuir com a criação de uma ferramenta mais completa para o aprendizado e livre de erros durante seu uso.
Este trabalho tem o propósito de expor todas as melhorias implementadas ao MACP e as experiências obtidas. E assim estimular que mais desenvolvedores possam engajar-se com o projeto MACP, tornando a ferramentas mais estável, completa e que atenda as necessidades da etapa de ensino a introdução a programação.

Palavras-chave: portugol. português estruturado. pseudo-linguagem. pseudocódigo.

ABSTRACT

SILVA, Leonardo. MACP a Free Software Experience in the Improvement of a Tool for Structured Portuguese. 2017. 34 f. – Curso de Ciência da Computação, Universidade Federal do Maranhão. São Luís, 2017.

One of the great difficulties for beginners in programming is the difficulty of assimilating the basic concepts related to problem solving using the algorithmic description in Portuguese (pseudo-codes). When using a simple language like Portugol, the need arises to create a tool that can execute the algorithms created from this formalism, making the learning process more practical.
It is very important to have a reliable tool available that meets the needs of users. Many developers make public source code for development. With this, people interested in the project can contribute in some way, either through corrections of identified problems or insertions of functionalities that can be useful in the learning process.
The proposal of a shared or free development approach causes several programmers to contribute by increasing in theory the speed at which the tool is developed where a greater number of eyes focused on the project can minimize the error rates in the version available for The users. The MACP today has its development efforts based on this free code policy so that programmers interested in the project can contribute to the creation of a more complete tool for learning and free of errors during its use.
This paper aims to present all the improvements implemented to the MACP and the experiences obtained. And thus encourage more developers to engage with the MACP project, making the tools more stable, complete and that meets the needs of the teaching stage the introduction to programming.

Keywords: portugol. portuguese structured. pseudo-language. pseudocode.

LISTA DE FIGURAS

Figura 1 – Modelo conceitual do processo 1
Figura 2 – Visualg imagem da ferramenta 5
Figura 3 – Portugol IDE 2.3 imagem da tela de início 6
Figura 4 – Portugol IDE 2.3 imagem da área de edição gráfica 7
Figura 5 – Portugol IDE 2.3 imagem do editor de texto 8
Figura 6 – Portugol Studio imagem da seção de inicio 9
Figura 7 – Portugol Studio imagem do editor de texto 9
Figura 8 – Portugol Studio imagem das bibliotecas 10
Figura 9 – Portugol Studio imagem da ferramenta de ajuda 11
Figura 10 – MACP imagem do editor de texto da ferramenta 12
Figura 11 – MACP Imagem das opções do menu executar 13
Figura 12 – MACP imagem da tela de exportação para a linguagem C++ 14

LISTA DE TABELAS

Tabela 1 – Quadro comparativo de Características 15

LISTA DE ABREVIATURAS E SIGLAS

C	Linguagem de progrmação procedural
C++	Linguagem de progrmação orientada a objetos
IDE	Integrated Development Environment
PHP	Linguagem de programação voltada para internet
VISUALG	Visualizador de Algoritmos
WEB	Sinônimo para rede mundia de computadores

SUMÁRIO

1 – INTRODUÇÃO 1
1.1 Justificativa 2
1.2 Objetivos Gerais 2
1.3 Objetivos Específicos 2
1.4 Delimitações 2
1.5 Organização 3

2 – TRABALHOS RELACIONADOS 4
2.1 VisuAlg 5
2.2 Portugol IDE 6
2.3 Portugol Studio 8

3 – MACP 12

4 – CONTRIBUIÇÕES INSERIDAS AO MACP 16
4.1 Abrir Arquivos 16
4.2 Salvar Arquivos 17
4.3 Configurações de Execução 17
4.4 Correções do Núcleo de Compilação do MACP 18
4.4.1 Expressões Booleanas 18
4.4.2 Expressões Aritméticas 19
4.4.3 Laço de Repetição para 20
4.5 Adição do Comando caso 22
4.6 Tradução do código Portugol para outra linguagem 24
4.7 Impressões durante o período de colaborador 25

5 – ANÁLISE E DISCUSSÃO DOS RESULTADOS 26

6 – CONCLUSÃO 31
6.1 Trabalhos futuros 32
6.2 Considerações finais 32

Referências 33

1 INTRODUÇÃO

Um dos desafios para quem está começando no mundo da programação é educar o pensamento de forma que todo o processo envolvido esteja bem definido, onde possam ser percebidos os passos necessários que conduzam até a resolução de determinada tarefa de forma algorítmica.

Analogamente, as etapas identificadas no processo de programação podem enquadrar as fases do processo de aprendizado em que o problema significa a busca pelo conhecimento, o desenvolvimento do algoritmo é o processo de aprendizado fazendo o uso de todas as ferramentas e métodos pertinentes , enquanto que a solução equivale ao conhecimento adquirido. A Figura 1 ilustra esse modelo conceitual.

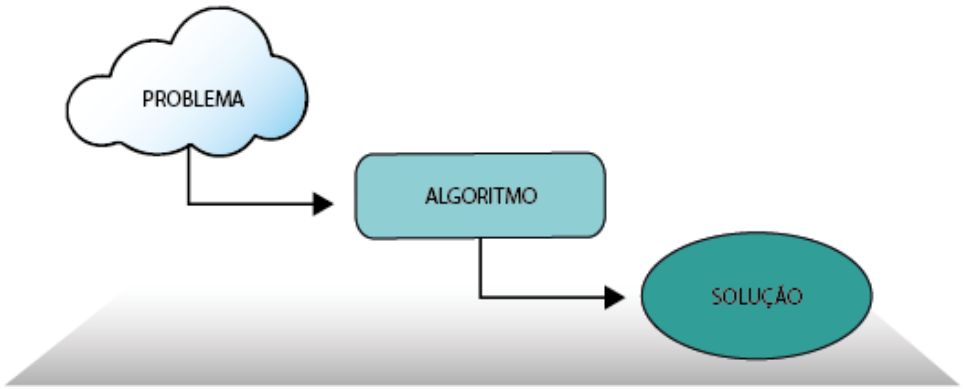

Figura 1 – Modelo conceitual do processo

Diante dessa situação, surgiu a necessidade de uma ferramenta que torne o processo de aprendizado mais efetivo e que faça uso de um formalismo que tenha semelhança com a linguagem nativa, pois o foco dessa etapa é a possibilidade da observação concreta de todos os conceitos e termos que são introduzidos.

Sabendo que desenvolver uma ferramenta de aprendizagem não é uma tarefa simples, e que seu propósito é puramente de cunho educacional, é que se percebe o quão benéfico é ter a possibilidade de poder agregar vários conceitos e ideias que possam tornar mais robusta, eficiente e com o mínimo de erros e falhas para o usuário final. Buscando contribuir com as diversas metodologias de ensino é que se tem uma gama de ferramentas como o MACP (Compilador Portugol), abordado no Capítulo 3. As funcionalidades podem ser desde a simples edição e execução de um programa escrito sob algum formalismo até a demonstração fluxográfica da estrutura do código desenvolvido, ampliando assim a perspectiva acerca das minúncias relativas ao conteúdo a ser aprendido.

Em relação às contribuições inseridas ao MACP, sob a perspectiva de desenvolvimento

colaborativo, o trabalho fornece, além de informações sobre cada uma das ferramentas, relatos sobre essa experiência de contribuição. Esses relatos estarão ratificando o quanto pode ser benéfico esse tipo de desenvolvimento, com a possibilidade da incorporação de conceitos que agreguem as funcionalidades resoluções melhoradas, além de revisões e otimizações para determinadas rotinas.

Tendo o objetivo de auxiliar o aprendizado, a prática e o dinamismo proporcionados pelo uso desse tipo de ferramenta é que se vê fortalecida a importância de se dedicar à construção de aplicações que possam ser usadas com essa finalidade. O MACP tem seu foco voltado para quem está ingressando nesse caminho.

1.1 Justificativa

O crescimento do número dos dispositivos e sua evolução constante abre espaço para novas áreas, técnicas e aplicativos de desenvolvimento resultando, assim, em uma demanda por profissionais capazes de criar soluções que atendam essas necessidades. Diante desse cenário, o uso de ferramentas que auxiliem a capacitação desse profissional torna essa tarefa mais eficiente à medida que todos os paradigmas propostos podem ser melhor assimilados com uso de uma boa aplicação que atenda às necessidades de cada etapa do aprendizado. Dessa forma, o presente trabalho visa apresentar as contribuições efetuadas em uma ferramenta, levando em conta a forma de desenvolvimento sob a qual o processo foi realizado.

1.2 Objetivos Gerais

Apresentar um relato das contribuições e das experiências no desenvolvimento de melhorias inseridas na ferramenta MACP com o propósito de melhorar seu uso e desempenho.

1.3 Objetivos Específicos

1. Apresentar uma descrição ilustrando entre as várias ferramentas analisadas, as características contempladas por cada uma expondo com isso a importância de seu uso
2. Contextualizar sobre a motivação e origem da ferramenta MACP
3. Fornecer informações sobre as vantagens e desvantagens no escolha do uso de software livre no contexto da ferramenta MACP.
4. Informar, baseado nos relatos de usuários, dados sobre a maturidade e portabilidade da ferramenta MACP, conduzindo-a sugestões para trabalhos futuros.

1.4 Delimitações

Este trabalho limita-se a trazer informações sobre alterações inseridas ao MACP, impressões obtidas na contribuição de melhorias do programa sob a política de software livre e de desenvolvimento colaborativo. Apresenta descrições comparativas entre as ferramentas

na ratificação da importância do seu emprego durante as fases iniciais de aprendizado de programação, tais como:

- Plataforma de desenvolvimento.
- Ambiente de execução.
- Características visuais.
- Funcionalidades existentes.
- Usabilidade

1.5 Organização

O presente trabalho é composto de 6 capítulos. No capítulo 1 é feita a introdução de todo trabalho, bem como seus objetivos conforme descrição seguinte.

Capítulo 2 – apresenta algumas descrições com ilustrações entre as várias ferramentas existentes contextualizando suas características de forma a reafirmar a importância de ter acesso a soluções que auxiliem no processo de aprendizado e assim justifique o seu uso como facilitador para exercício e fixação de conhecimento.

Capítulo 3 – demonstra informações sobre histórico e motivações relativas ao MACP, contendo descrições de telas e suas funcionalidades que podem agregar benefícios para o usuário, trazendo também impressões sobre a forma de suporte a qual a mesma possui.

Capítulo 4 – nele estarão descritas todas as melhorias inseridas ao MACP para melhorar seu desempenho e solucionar problemas existentes em precedimentos como o de identificar configurações de ambiente para executar, abrir e salvar o código fonte[1], além da adição de um novo recurso para a linguagem, o comando "caso", levando sempre em consideração a experiência como colaborador.

Capítulo 5 – faz uma análise a partir de exemplos, criados com a função de testar e demostrar as modificações inseridas exibindo o padrão de codificação dos comandos usados em cada exemplo.

Capítulo 6 – informa sobre as conclusões obtidas durante a fase de desenvolvimento, teste e implantação das modificações, além de sugestões de melhorias para trabalhos futuros.

[1] Conjunto de palavras descrevendo instruções obedecendo um formalismo em uma linguagem de programação

2 TRABALHOS RELACIONADOS

As formas de representação de algoritmos mais conhecidas são descrição narrativa, descrição por fluxograma e a convencional descrição por pseudocódigo. A descrição narrativa ocorre com o emprego da linguagem natural, usada no dia a dia pelas pessoas para enunciar uma sequência cronológica de fatos a serem seguidos. A descrição fluxográfica faz uso de formas geométricas para representar ações distintas, de modo geral, facilita a compreensão das ideias contidas. Ao passo que a representação por pseudocódigo possui uma riqueza de detalhes, que lhe deixa próximo de uma linguagem de programação convencional, característica pertinente e que lhe dá a condição de grande aceitação, devido a possuir um bom nível de formalização que é suficientemente geral para permitir a tradução direta para uma linguagem de programação.

Baseado no razoável rigor que qualifica o pseudocódigo, também chamado de linguagem estruturada, é que se iniciou o desejo por uma ferramenta que fosse capaz de executar o Português Estruturado. Porém, não há um consenso sobre a grafia das instruções, e tão pouco da estrutura formal de cada instrução, levando a um cenário onde cada ferramenta criada possui uma gramática distinta. O que causa uma certa nebulosidade na decisão por qual ferramenta escolher.

É importante ressaltar que essas ferramentas se diferenciam dos softwares de trabalho profissionais que têm como meta principal alcançar uma alta produtividade e que os fatores de usabilidade são idealizados de maneira a contribuírem o máximo possível com a utilização do usuário, promovendo que se tenha o menor esforço (GURGEL, 2006).

A concepção de uma ferramenta com qualidade deve ser pautada na eficácia de realizar corretamente as operações para a qual foi criada, de modo que realizado as ações certas para resolver uma tarefa se tenha uma garantia no resultado, atingindo a eficiência. E assim os principais aspectos para ter um bom resultado é a aparência da interface.

Quando se pensa em interação usuário máquina, a interface é um dos elementos mais importantes para que essa relação seja boa de fato, pois os elementos que a compõe, quando bem expostos, quanto a sua função, facilitam a utilização da ferramenta. A qualidade de uma ferramenta deve ser mensurada do ponto de vista do usuário e a interface é a parte do software com o qual ele interage. Nesse contexto da criação de software, o enfoque deve considerar o usuário como um dos centros do processo. A interface deve agir como facilitador devendo sempre objetivar promover uma maior integração entre os conceitos teóricos e a prática, ampliando o horizonte do aprendizado.

Na pesquisa realizada, foram encontrados alguns trabalhos relacionados ao uso de aplicações para auxiliar o aprendizado dos fundamentos introdutórios de programação. E com base nesse conteúdo serão apresentadas informações como plataforma em que foi desenvolvida, em quais ambientes podem ser executados, autor, se ainda possui suporte e funcionalidades

apresentadas.

2.1 VisuAlg

Sousa (SOUZA, 2016), desenvolveu a ferramenta chamada VisuAlg (Visualizador de algoritmos) [1] um programa de livre uso e distribuição, empregado no ensino de programação em várias escolas e universidades no Brasil e no exterior. Dentre os recursos estão as possibilidades de editar, interpretar e executar algoritmos em pseudo-código, características muito úteis para auxiliar o aprendizado.

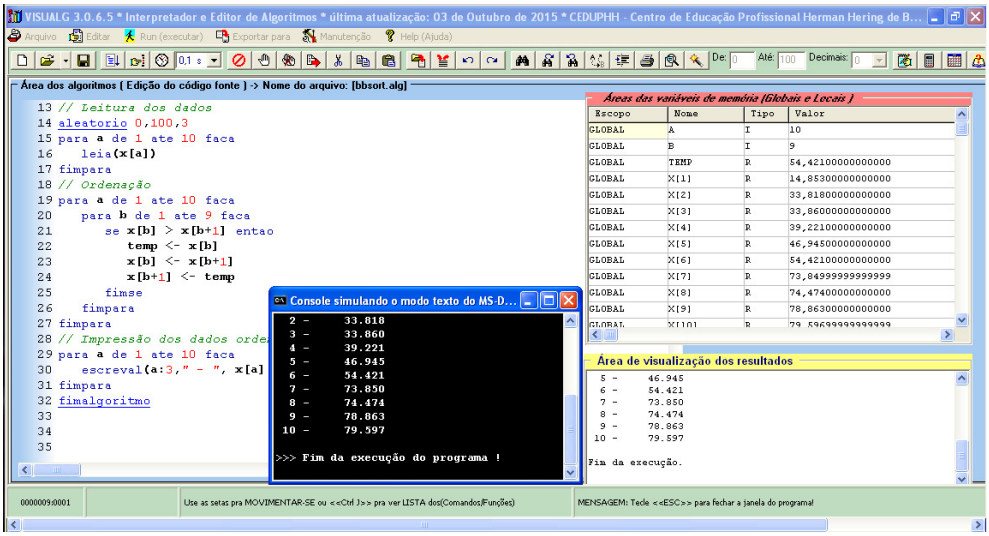

Figura 2 – Visualg imagem da ferramenta

O suporte à ferramenta se faz através dos canais de contatos disponíveis no sítio de distribuição. O VisuAlg encontra-se desenvolvido especificamente para sistemas Windows, concebida em plataforma de programação Delphi e com uso dificultado por usuários de ambientes Linux e Mac.

A aplicação conta com uma área de visualização de valores das variáveis de memória global e local, sendo assim um item importante quando associado à execução do algoritmo através do recurso de temporização da velocidade em que o mesmo executa, e assim facilitando o acompanhamento de cada instrução a ser processada, melhorando o entendimento de cada parte do código. Outro recurso importante e que faz uso desse campo é a rotina para depuração, onde cada ponto de interrupção é uma oportunidade de se checar os valores das variáveis usadas e assim avaliar se o que está sendo executado é de fato o que se espera. Possui um guia de ajuda e de informações contendo exemplos das instruções da linguagem e histórico da aplicação.

Além das operações de salvar e abrir aquivos, a ferramenta conta com a funcionalidade de exportar o código fonte escrito em Portugol para a linguagem Pascal, um editor que já

[1] http://www.apoioinformatica.inf.br/produtos/visualg/

apresenta um esqueleto de código com a intenção de poupar trabalho ao usuário e mostrar o formato básico a ser seguido, palavras reservadas com sintaxe em destaque, comentários direto em código e auxílio para auto-digitação com uso de atalhos definidos na aplicação.

O ambiente de codificação conta ainda com uma área de visualização de resultados e um console de interação com o usuário fornecendo opções de entrada e saída ampliando, com isso, a experiência e estimulando a capacidade de criar rotinas mais lapidadas.

2.2 Portugol IDE

Idealizado como uma ferramenta de apoio ao ensino de programação de Cursos de Engenharia da Escola Superior de Tecnologia de Tomar do Instituto Politécnico de Tomar (MANSO; OLIVEIRA; MARQUES, 2009) equipado com um conjunto de ferramentas pedagógicas para a aprendizagem de técnicas de programação, o interesse é fazer com que o aluno mantenha o foco no desenvolvimento do algoritmo e assim proporcionar uma melhor transição para os ambientes de programação tradicionais. Conta com uma boa documentação sobre a ferramenta e sobre a linguagem disponíveis no sítio de armazenamento e na própria ferramenta.

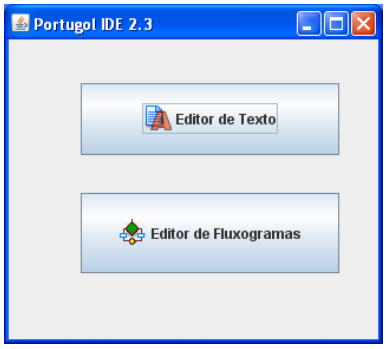

Figura 3 – Portugol IDE 2.3 imagem da tela de início

O Portugol IDE[2] é um simulador de linguagem algoritmica desenvolvido para apoio às aulas de introdução à programação. Possui uma linguagem simples e minimalista quanto ao número de palavras reservadas, porém bastante expressiva e abrangente quanto aos recursos disponíveis. Uma característica no início da execução da mesma é a possibilidade de seleção por qual modo o aluno deseja desenvolver, se de forma textual ou de forma gráfica, como mostrado na Figura 3. Podendo também alternar entre os modos e salvar o trabalho feito tanto em forma de texto quanto de forma fluxográfica. Nesse caso o texto poder ser convertido para forma gráfica e vice-versa.

A possibilidade de construir algoritmos de forma gráfica deve possibilitar o aluno atingir seus objetivos de aprendizagem de forma adaptável (MORENO; MYLLER, 2003) para que sua compreensão acerca do conteúdo passado possa ser melhor assimilado e que o entendimento sobre as instruções a nível de código fonte seja o mais claro possível. Dessa forma, os passos

[2] https://github.com/iptomar/portugol

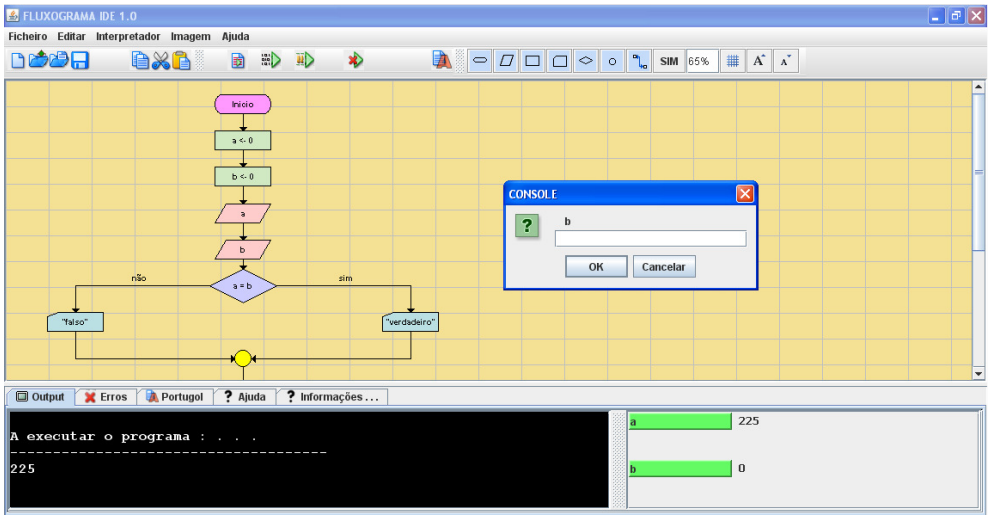

Figura 4 – Portugol IDE 2.3 imagem da área de edição gráfica

cognitivos necessários para o usuário desempenhar suas tarefas de maneira completa usando a interface devem estar bem dimensionados para que as ações mentais e físicas estejam direcionadas com a execução e resolução da tarefa sem dispersões ocasionadas por problemas de comunicação com a interface (CARROLL, 1991).

O editor gráfico de algoritmos, visto na Figura 4, permite construir, executar e visualizar a tradução para o Portugol das instruções fornecidas. Entre os elementos existentes, há indicador de início, fim, conectores concentradores e indicadores de fluxo, estruturas condicionais, estruturas de entrada e saída, declaração de variáveis, um console de interação para casos de entrada do usuário e uma área para acompanhamento de valores de variáveis durante uma execução de forma depurada.

A ferramenta conta com um editor de texto completo, observado na Figura 5, com destaque para o realce de cor da fonte, compilação, execução normal e monitorada, transição para visualização fluxográfica da estrutura algoritmica do código fonte, guia de ajuda para estrutura e sintaxe da linguagem, guia de auxílio sobre a ferramenta, guia de exibição de erros e para saída do console, além de opção para formatação automática da estrutura do código. Possui sua execução facilitada em outros ambientes decorrentes do seu desenvolvimento ser em Java, ficando assim portável (WIKIPEDIA, 2017)[3], pois sua dependência fica por conta da instalação da máquina virtual Java e não quanto da arquitetura do sistema operacional.

O Portugol IDE possui embutido os painéis de ajuda e de informações contendo instruções sobre como fazer uso dos comandos pertinentes às estruturas sintáticas da linguagem, ambiente e versão da aplicação.

Há uma gama de literatura relacionada às dificuldades da inserção dos conceitos de programação, indicando que esse assunto é difícil, no aspecto de aprendizado e ensino. Há um consenso geral da abordagem quanto aos cursos de programação de nível introdutório que

[3]Capacidade de um programa ser compilado ou executado em diferentes arquiteturas independente de hardware ou de software

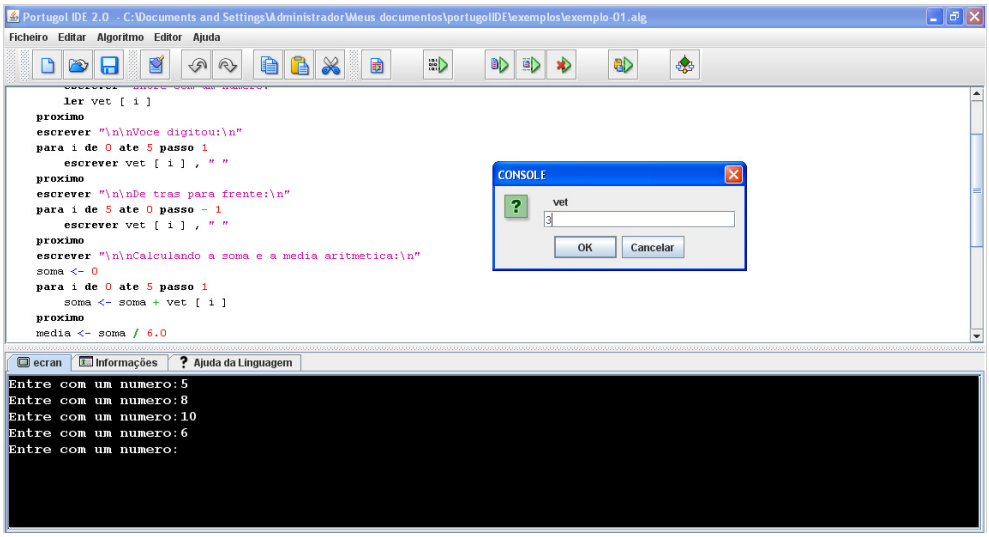

Figura 5 – Portugol IDE 2.3 imagem do editor de texto

têm tradicionalmente elevadas taxas de reprovação (BUTLER; MORGAN, 2007). Tendo ainda autores que frisam que essa é uma característica geral, não sendo restrito a um curso, uma instituição ou grupo de discentes de perfil específico (GIANGRANDE, 2007) .

As ferramentas que fazem uso de representações gráficas durante a execução de algoritmos podem permitir aos alunos vários níveis de interação que vão desde a ratificação das soluções construídas a retificação dos erros, além de possibilitar o emprego de novos caminhos para solucionar um problema (GOMES; MENDES, 2000).

2.3 Portugol Studio

Todo e qualquer software que possa ser usado com algum objetivo educacional, pedagogicamente aceito e defensável, tanto por alunos e professores, independente do objetivo para o qual ele foi criado é considerado um software educativo (TEIXEIRA; BRANDAO, 2003).

Outra ferramenta com esse propósito é o Portugol Studio[4], uma IDE (Integrated Development Environment)[5] projetada com finalidade didática podendo assim reduzir os problemas decorrentes do aprendizado e ensino, que permite que o tempo e a energia do estudante estejam voltados na aprendizagem da lógica propriamente dita (BEAUBOUEF; MASON, 2005).

Em sua janela inicial, apresentada na Figura 6, a ferramenta conta com uma área com diversos exemplos distribuídos em categorias servindo como material de apoio para o entendimento de sua sintaxe, além de disponibilizar diversos atalhos para funcionalidades com ajuda, bibliotecas, atualizações, informações sobre a ferramenta e possibilidade da criação de programas mais complexos como jogos, possuindo ainda uma quantidade razoável de participantes ativos mantedores do projeto.

[4] http://lite.acad.univali.br/portugol/
[5] Ambiente de Desenvolvimento Integrado

Figura 6 – Portugol Studio imagem da seção de inicio

Sua construção possui diversos elementos presentes em ferramentas de cunho profissional, mas sempre mantendo o foco em ser uma ferramenta de auxílio a pessoas que estão começando no mundo da programação. Assim, facilitando o manuseio por parte do usuário, pois sua interface possui elementos bem destacados que auxiliam na memorização, devido à forma de organização adotada, para os elementos interativos da interface gráfica.

Dentre suas características, o editor possui colorização em destaque para elementos do código fonte, console de entrada e saída, árvore estrutural que permite identificar e compreender os elementos que compõem o programa escrito, através de símbolos com informações de tipos de variáveis e de retorno de rotinas, uma região para observar o estado das variáveis durante o processo de depuração e uma área para mensagens com o objetivo de auxiliar através de dicas a identificação dos problemas que estiverem presentes durante a codificação.

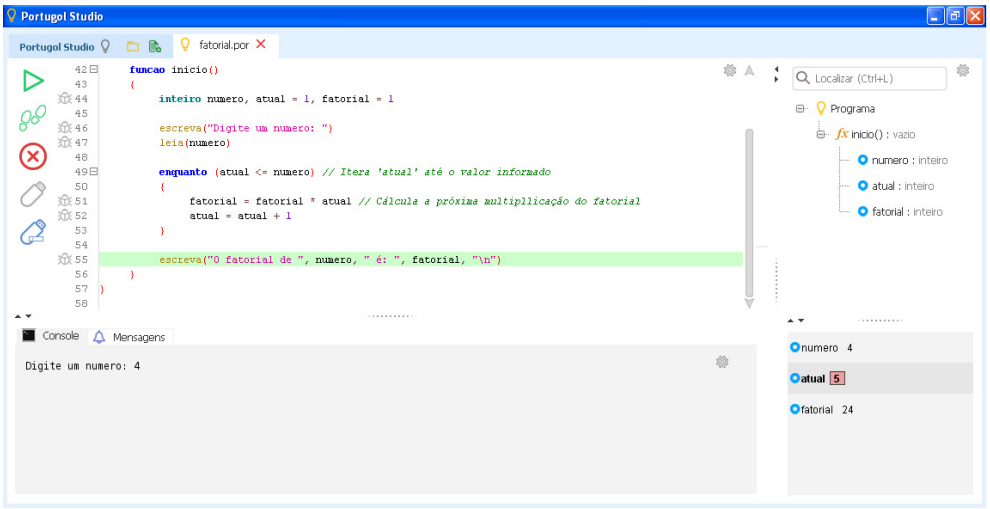

Figura 7 – Portugol Studio imagem do editor de texto

Pesquisas apontam que a maioria dos livros utilizados como referência básica nas

disciplinas de introdução a programação em universidades brasileiras possuem explicações usando alguma notação de Portugol (ZANINI; RAABE, 2012). Isso corrobora a importância de se ter aplicações que possam interpretar o Portugol que, para usuários de língua portuguesa, facilita o entendimento acerca do estudo de programação.

Um sistema computacional com finalidades educacionais não pode ser feito sem considerar o seu contexto pedagógico de uso. No MACP, o foco deve sempre tentar atender as características das interações de ensino e aprendizado, permitindo que o usuário possa evoluir e consolidar as teorias recebidas.

Uma aplicação concebida como software educativo e interativo, intencionalmente criada para facilitar a aprendizagem de conceitos específicos. Pode ter classificações como softwares de tutoriais, exercícios e práticas, programação, simulação e jogos (VALENTE, 1993; VIEIRA, 2005). Visto dessa forma, a construção de software de auxílio ao aprendizado pode contribuir com muitos ramos de estudo.

As bibliotecas do Portugol Studio fornecem estruturas que facilitam a resolução de alguns problemas onde parte da solução exige o resultado de um objeto complexo que necessariamente não precisa ser abordado nas primeiras etapas de estudo. Permitindo assim que possa haver a escrita de programas um pouco mais complexos incluindo jogos sem ter um grande aprofundamento. Podendo assim beneficiar ainda mais o interesse do aluno acerca da atuação no ramo do desenvolvimento de aplicativos.

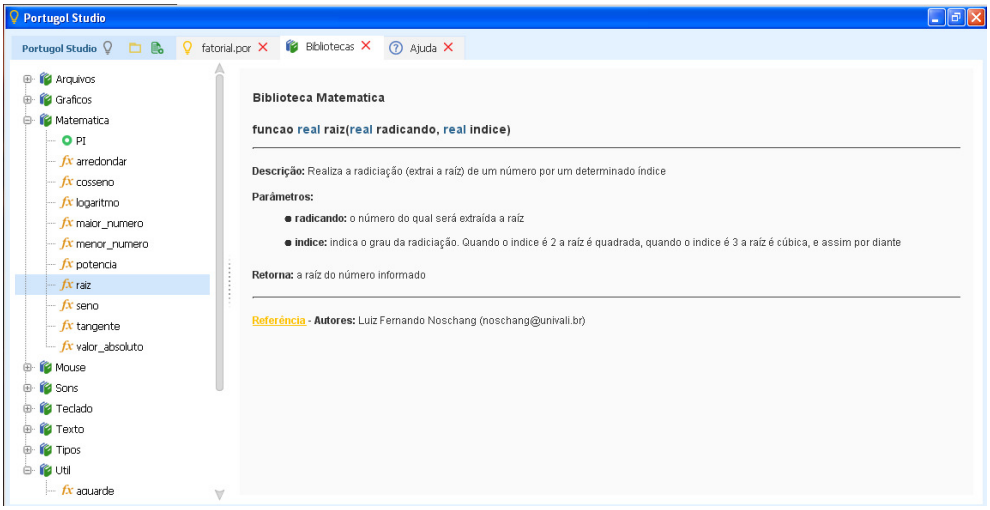

Figura 8 – Portugol Studio imagem das bibliotecas

Para pessoas que estejam iniciando os estudos sobre construção de programas a ideia da linguagem Portugol é facilitar a compreensão sobre esse processo. E com isso, não se torna tão relevante introduzir de cara uma complexidade de conceitos que estão presentes em formalismos de linguagens de cunho profissional, como C, C++, Java criando uma resiliência conceitual ocasionadas pela não abstração dos elementos que compõe esses formalismos, deixando espaço para uma melhor fundamentação dos objetos teóricos característicos dessa fase do aprendizado.

Outra característica presente no Portugol Studio é o seu conteúdo de ajuda embutido na própria ferramenta melhorando a experiência do usuário pois não há a necessidade de utilizar um outro mecanismo para ter auxílio seja sobre a sintaxe da linguagem ou sobre a ferramenta em si.

O acesso às informações da ajuda, conforme Figura 9, é facilitado devido a organização por tópicos e em estrutura de árvore com o conteúdo sucinto, direto e prático. Existem outros canais de obter auxílio que vão de vídeos a contatos com pessoas envolvidas com o projeto ou no próprio sítio de disponibilização da ferramenta que já foi referência de vários trabalhos relacionados a temática do uso da linguagem Portugol como facilitador do aprendizado.

Figura 9 – Portugol Studio imagem da ferramenta de ajuda

A melhoria da qualidade da educação deve acompanhar as inovações curriculares e didáticas, não deixando de usufruir dos recursos e inovações disponíveis que possam contribuir efetivamente nos temas de matéria educativa (MERCADO, 2002).

O emprego de tecnologias na educação servem de suporte em diversas modalidades de ensino, como o ensino presencial e a distância, além de facilitar à auto-aprendizagem. A abordagem do tema em pesquisas e por autores, demonstram que uso de software educativos como um meio didático ou como ferramenta de ensino, contribui expressivamente para práticas acadêmicas em qualquer nível de ensino. E sob este aspecto é que se observa que as ferramentas que processam o Portugol procuram adquirir características que visam ter uma atuação adequada no processo de interação com os usuários, adotando conceitos visuais que agreguem significados a partir de suas funcionalidades.

3 MACP

O MACP surgiu como um projeto[1] com a finalidade curricular avaliativa durante o período de graduação em 2006 na Universidade Federal do Maranhão para disciplina de compiladores que contava inicialmente com os desenvolvedores Daniel Lima Gomes Júnior, Leandro de Sousa Marques e Ulysses Santos Sousa. Fomentou-se então a possibilidade de utilização da mesma para auxílio dos discentes, que iniciavam o aprendizado de programação, e que sentiam dificuldades quando o conteúdo ficava restrito apenas no campo teórico ou que a prática limitava-se a descrever e simular de forma manual. Isso tornava esse processo custoso quando a rotina exigia passos mais complexos e sobre um conjunto de possibilidades mais amplo, podendo em alguns casos, dependendo da metodologia do docente, que ministra a disciplina, fazer o uso de uma linguagem profissional, a exemplos C e Pascal cujo rigor dos formalismos podem tornar de início o processo mais difícil.

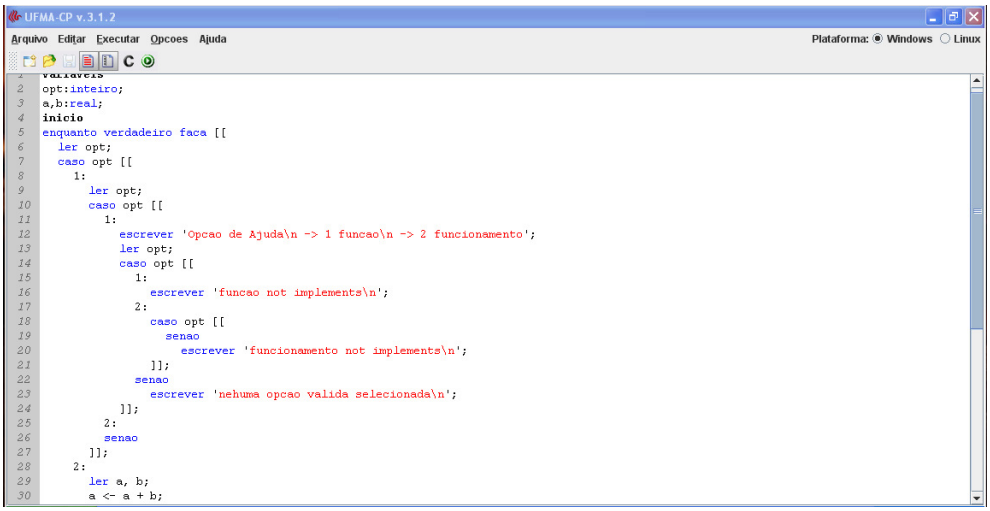

Figura 10 – MACP imagem do editor de texto da ferramenta

Em um trabalho realizado com alunos dos cursos de ciências exatas da Universidade Federal de Roraima (UFRR), de Introdução à Ciência da Computação e Cálculo Numérico, apontou que uma grande dificuldade dos alunos está no entendimento do problema proposto e com a capacidade de organizar o raciocínio logicamente afim de encontrar a solução. Isso ainda é um agravante que alguns alunos sentem uma maior dificuldade com a introdução da língua inglesa, que compõem as linguagens de programação com suas mensagens de erro como forma de apoio ao desenvolvimento de algoritmos (SOUZA, 2013).

O MACP conta com um editor com realce de cor da fonte, que ajuda na identificação dos elementos que compõem o código fonte, possui numeração de linhas, botões para execução, visualização do código fonte em C++. Está presente o recurso para o usuário realizar a análise

[1] http://portugol.sourceforge.net/

léxica e análise sintática. Assim dando a possibilidade de observação, no caso da análise léxica, observar todos os tokens[2] mapeados a partir de uma implementação de Portugol realizada pelo usuário.

É na análise sintática, que é feita a verificação da estrutura gramatical a partir dos termos fornecidos pela análise léxica e com isso identificando se as regras de geração da gramática estão sendo atendidas. A exportação do código fonte elaborado pelos alunos em Português Estruturado (Portugol) está presente com o objetivo de facilitar a ambientação e o aprendizado de uma linguagem profissional que possivelmente possa ser abordada em etapas posteriores do curso.

Devido o interesse de pessoas que enviaram e-mails para obter mais informações sobre a utilização da ferramenta decidiu-se criar um projeto, que atualmente encontra-se no sítio SourceForge[3], fornecendo assim mais uma opção de ferramenta para que usuários possam desenvolver suas soluções diretamente em Portugol.

O conteúdo do projeto MACP está disponível na WEB para acesso, cópia e modificação. Permitindo que pessoas interessadas, possam fazer alterações ou atuar de forma a contribuir com o projeto. Podendo assim, criar melhorias e recursos que possam ser úteis dentro da proposta a qual o projeto MACP está inserido, levando assim, para o usuário final, uma ferramenta mais equipada para atender as necessidades que possam surgir durante seu uso, como uma ferramenta de auxílio ao aprendizado.

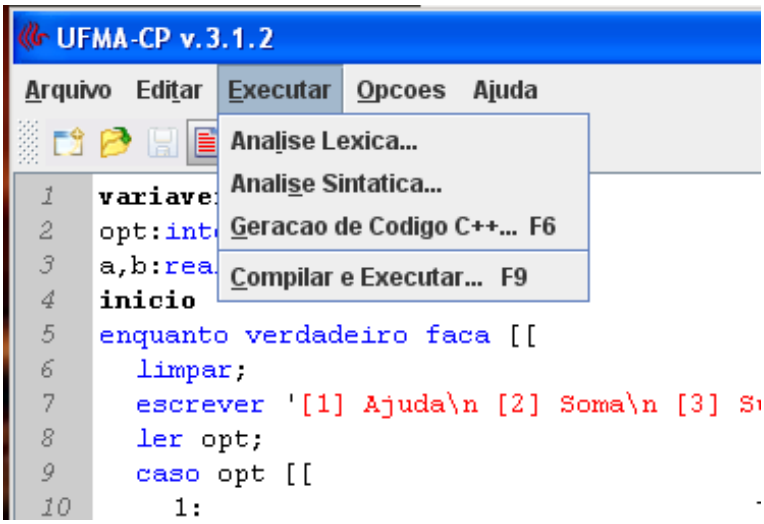

Figura 11 – MACP Imagem das opções do menu executar

Todo o projeto tem seu código fonte desenvolvido em linguagem de programação Java, onde a geração do binário final tem seu uso facilitado por usuários dos mais variados tipos de sistemas operacionais, devido ao fato da maquina virtual, que é necessária, ser multiplataforma tornando o seu uso possível com poucas ou nenhuma modificação. Com essa característica

[2] Sequência de símbolos extraídas do código fonte.

[3] Repositório de código fonte para desenvolvedores gerenciarem projetos livres e de código aberto colaborativamente

vinculada tem-se uma vantagem advinda da flexibilidade do seu uso por usuários dos mais variados nichos, aos quais já estão familiarizados e portanto não a necessidade de adicionar mais elementos que possam vir adicionar entraves na sua utilização.

O MACP fornece informações sobre a gramática que utiliza, exporta seu código fonte para outras linguagens de programação conforme observado na Figura 11 e 12, fornece opções para realizar analise léxica e sintática independentes, permitindo assim a visualização dos possíveis tipos de erros que podem ocorrer durante a construção do algoritmo, facilitando o aprendizado.

Após a abordagem do estudo sobre as teorias de programação, feitas pelo educador, este terá que indicar, de acordo com a forma que apresenta o conteúdo teórico, o software que ele acredita ser capaz de ajudar e estimular o processo do aprender, que o caracterizará como software educativo (OLIVEIRA; COSTA; MOREIRA, 2001).

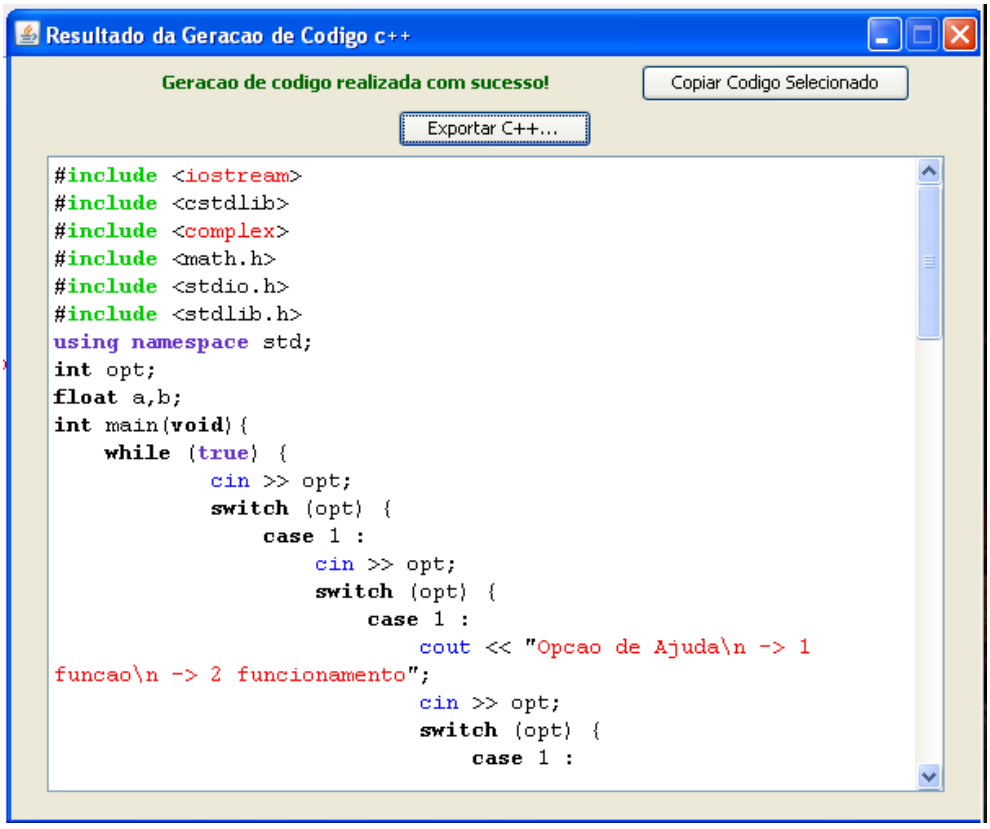

Figura 12 – MACP imagem da tela de exportação para a linguagem C++

A vantagem da execução do algoritmo em um ambiente simples e familiar maximiza a compreensão das abstrações apresentadas, pois faz-se uso de recursos ao qual o aluno já conhece (PATTIS; ROBERTS; STEHLIK, 1995). Todavia a execução e representação do algoritmo num ambiente diferente do qual aluno esteja engajado poderá resultar em que este não aprenda conceitos básicos como, por exemplo, o conceito de atribuição, variável e de laços de repetições.

Observando os argumentos de vários profissionais tem-se um grande apanhado de

evidencias do quão benéfico é o emprego de ferramentas que ajudem a fixar o conhecimento através da resolução de execícios em ambientes propícios para tal. Então ver-se a importância da existência de aplicações como o MACP em que o seu desenvolvimento e evolução esteja baseado no auxílio a aprendizagem, fornecendo recursos que possam facilitar esse caminho.

As disciplinas de Algoritmos e Programação são componentes curriculares obrigatórios do primeiro período de diversos cursos de ciências exatas e das áreas tecnológicas. Sendo que essas disciplinas acumulam uma bagagem de altas taxas de reprovação e evasão. Identificar e entender os mecanismos que tornam essa realidade bastante presente no ensino e aprendizado dessas disciplinas se torna fundamental para que se encontre uma forma de contornar essas adversidades.

A partir das adversidades encontradas nas atividades de ensino e aprendizado de algoritmos é essencial que o docente busque o aperfeiçoamento constante de suas técnicas e metodologias de ensino. Pois o uso de uma ferramenta adequada melhora a memorização de conceitos abordados, uma vez que esta sempre que detectar erros durante a resolução de problemas com seu uso obrigará o aluno a voltar para o ponto em que foi identificado o erro maximizando o esforço na obtenção da resposta correta tirando a sua preocupação com o processo.

A escolha de uma maneira de apresentar o conteúdo e das ferramentas utilizadas deve ser considerado durante o planejamento o perfil dos alunos. É razoavelmente comum ter como desafios para o docente um número elevado de alunos por turma, falta de compreensão por partes dos alunos da necessidade das disciplinas, enquanto que para o discente há dificuldades de desenvolver o raciocínio lógico com abstração para uma linguagem de programação (QUINTELA; RIBAS, 2016) relacionado muitas vezes ao baixo nível de conhecimento do inglês.

A adoção de uma ferramenta de suporte como o MACP pode incrementar positivamente os processos de ensino e aprendizado, pois a utilização das tecnologias existentes dentro da sala de aula e fora dela, é uma forma de estender o ambiente de aprendizado para um ambiente virtual, que venha a favorecer a interação entre professor e aluno, sendo o termo, interação considerado uma palavra-chave dos tempos atuais.

A seguir tem-se apresentação da Tabela 1 mostrando algumas características contempladas por cada ferramenta onde, de acordo com a didática e perfil dos alunos a sua adoção, pode facilitar a sua escolha como uma ferramenta de suporte ao aprendizado.

FERRAMENTA	MULTIPLATAFORMA	EDIÇÃO/VISUALIZAÇÂO GRÁFICA	EXPORTA PARA OUTRA LINGUAGEM	DEPURAÇÃO	LINGUAGEM DE DESENVOLVIMENTO
VISUALG	NÃO	NÃO	SIM	SIM	DELPHI
PORTUGOL IDE	SIM	SIM	NÃO	SIM	JAVA
PORTUGOL STUDIO	SIM	NÃO	NÃO	SIM	JAVA
MACP	SIM	NÃO	SIM	NÃO	JAVA

Tabela 1 – Quadro comparativo de Características

4 CONTRIBUIÇÕES INSERIDAS AO MACP

A Cada ciclo de desenvolvimento é importante que haja uma avaliação empírica que possa abordar as necessidades e os problemas em que a próxima versão em desenvolvimento deve se concentrar. Isto significa que a avaliação não deve ser apenas o fim do desenvolvimento, mas também deve ser considerada como o começo de cada novo ciclo de desenvolvimento.

Este capítulo apresenta as contribuições específicas no projeto MACP. Situa-se como uma relato de experiência da participação em um projeto de software livre.

4.1 Abrir Arquivos

Na melhoria da rotina de abrir arquivos foi adicionado um filtro que tem como objetivo facilitar o reconhecimento durante a navegação dos arquivos que podem ser abertos para edição pelo MACP.

As operações para abrir arquivos foram revisadas para que fosse corrigido o problema que ocasionava a perda do trabalho que estava sendo realizado conforme Listagem 4.1, devido a ação do usuário para abrir um novo arquivo, pois esse evento fazia a substituição por um novo conteúdo sem exibir um diálogo com o usuário, descartando todo o código fonte que estava em edição até então, impossibilitando que com isso pudesse ser realizada as medidas cabíveis pelo usuário, de forma que sua escolha reflita a sua ciência no resultado final para contornar o impasse.

```
    Compilacao.rec = arquivoo = abrirArquivo.getSelectedFile();
    if(arquivoo != null && arquivoAux != null){ //
        int op = JOptionPane.showConfirmDialog(this,
                    "Deseja abrir um outro arquivo ?",
                    UfmaCp.version, JOptionPane.OK_CANCEL_OPTION,
                       JOptionPane.QUESTION_MESSAGE);
        if(op == JOptionPane.CANCEL_OPTION){
                return;
        }
    }
```

Listing 4.1 – Verifica se o usuário deseja abrir outro arquivo

A verificação da ação é efetuada através da solicitação de confirmação para o usuário, uma janela é apresentada e nesse momento questionando o prosseguimento da tarefa, fornecendo assim, a opção de anular a abertura de um outro arquivo e assim ter o texto do código fonte na área de edição preservado. O trecho de código da Listagem 4.1 apresenta as instruções inseridas para controle de abertura de um outro arquivo caso já exista um em edição, evitando perdas.

4.2 Salvar Arquivos

Uma melhoria adicionada conforme Listagem 4.2, foi a verificação da existência de um arquivo com o mesmo nome, fornecendo para o usuário uma mensagem de retorno requerendo sua atenção, e assim a continuidade dessa ação não será concretizada sem o seu consentimento.

Um outro problema identificado pelos usuários do MACP foi a presença de uma oscilação no momento de salvar, o arquivo não era criado e assim o texto escrito não era salvo, e esse problema não retornava nenhuma mensagem para dar conhecimento dessa falha para o usuário, impactando em alguns momentos na perda do texto codificado. A verificação da existência trouxe o benefício da confirmação da criação do arquivo em disco ou que ele ainda precisa ser gravado.

```
if (arquivoo.exists()) {
    int response = JOptionPane.showConfirmDialog(Interface.this,
                   "Esta pasta ja contem um arquivo chamado " +
                      arquivoo.getName()
                    + "\nDeseja substituir o arquivo existente? ",
                        "Confirmar substituicao de arquivo",
                    JOptionPane.OK_CANCEL_OPTION, JOptionPane.
                       QUESTION_MESSAGE);

    if (response == JOptionPane.CANCEL_OPTION) {
            return;
    }
```

Listing 4.2 – Verificação de existência de arquivo

A Listagem 4.3 apresenta uma verificação que detecta a falta da extensão. Caso ela exista, é criada uma nova instância para o arquivo adicionando a extensão adequada, evitando uma falha relacionada com a criação do arquivo, pois em alguns momentos o arquivo era criado em disco sem ter uma especificação de extensão, que auxilie seu manuseio e dessa forma deixava o seu reconhecimento e localização confusos.

```
if (!arquivoo.getName().endsWith(".txt")) {
        arquivoo = new File(caminho + ".txt");
```

Listing 4.3 – Verificação de extensão de arquivo

4.3 Configurações de Execução

Entre os requisitos para que o código fonte escrito em Portugol possa ser traduzido para uma linguagem executável, é preciso que seja feita a edição do arquivo de configuração do MACP. É por meio das informações de configuração fornecidas que as estruturas internas

podem ter suas instruções realizadas, que são essenciais nas operações de geração do código executável e por fazer sua execução.

Em relação às configurações, foi criada uma forma de obter todos os parâmetros necessários à execução das rotinas através das variáveis de ambiente, que são de controle do sistema operacional, e que contém as informações de localização das aplicações que são instaladas nesses ambientes, podendo ter seu conteúdo editado diretamente pelo usuário.

Com essa alteração, se o método para configurar a aplicação através de seu arquivo padrão de configurações não for bem sucedido, a alternativa de usar as variáveis de ambiente podem trazer uma maior eficácia, pois as variáveis de ambiente tem seu conteúdo reconhecido facilmente pelo sistema operacional, que pode fazer uso dessa informação para suprir os requisitos de funcionamento do MACP.

4.4 Correções do Núcleo de Compilação do MACP

A análise léxica, sendo a primeira fase do processo de compilação, é responsável por reconhecer e representar o código fonte em uma lista de tokens, elementos do código fonte, que são usados na verificação da estrutura gramatical pela análise sintática.

Na análise sintática, os tokens são avaliados quanto às regras de geração definidas pela gramática da linguagem Portugol do MACP. No entanto, a partir de relatos de usuários do MACP foram detectadas falhas no processamento de instruções previstas e que deveriam estar suportadas. Outras inconsistências identificadas eram relacionadas a comandos descritos que não tinham suporte implementados. A listagem 4.4 a seguir mostra o trecho para gravar o arquivo de execução.

```
try {
    PrintStream p = new PrintStream(exeFile);
    p.print(cppRefecence);
    p.close();
    PrintStream b = new PrintStream(exeBat);
```

Listing 4.4 – Trecho do código para gravar o arquivo de execução

4.4.1 Expressões Booleanas

Na construção de expressões lógicas iniciadas com uma negação na sentença ou com as constantes para verdadeiro e falso, que são palavras reservadas da linguagem, esses comandos não eram reconhecidos quando empregados pelos usuários em seus códigos de programação, resultando em erros durante a tradução.

O uso de expressões aritméticas na composição das instruções condicionais, contidas no código fonte desenvolvido pelo usuário, apresentava erros quando submetidas às regras de formação do formalismo que define a linguagem usada pelo MACP. As implementações para

gramática do Portugol do MACP não contemplava a totalidade dos recursos da linguagem que eram ditos suportados pela ferramenta.

A seguir, a Listagem 4.5 apresenta o trecho de código usado para suprir a necessidade de fazer uso de expressões lógicas inciadas com a negação do seu resultado. A instrução de negação, sendo uma palavra reservada, possui um valor numérico inteiro, assim como as demais, facilitando a comparação, pois estas marcas da linguagem vêm identificadas com seus respectivos códigos da fase de tokenização, a análise léxica.

```
switch (tk.getClas()) {
        case 43: {//
            codigoGerado += " !";
            break;
       }
```

Listing 4.5 – Trecho com a codificação da correção para negação no incio de expressões

A comparação entre uma variável e uma constante lógica como verdadeiro ou falso foi corrigida adicionando uma regra de formação para a gramática dentro da estrutura reponsável por fazer a análise das expressões que usam uma variável declarada no escopo do algoritmo.

E nessa parte, que é uma ramificação da árvore de formação da gramática do MACP, após encontrar um token, do tipo identificador, permite que o outro termo da expressão possa ser uma palavra reservada que represente uma valor lógico booleano. A Listagem 4.6 apresenta o trecho de código adicionado e que faz referência a um método, que já existia, para permitir o uso dessa combinação de elementos numa expressão condicional.

```
err = termoLogico();
if (!"".equals(err)) {
    return erro("Expressao logica parte invalida");
}
```

Listing 4.6 – Trecho da correção para uso de constantes lógicas

4.4.2 Expressões Aritméticas

O emprego de expressões aritméticas para compor formações lógicas eram restritas a comandos que possuíam parênteses em sua estrutura, assim o seu emprego em outras formações, como laços de repetições, que eram considerados erros. É que para outros comandos não há a obrigação de se usar parênteses, logo quando as expressões aritméticas eram usadas. Exemplo disso, é seu uso para a construção de algum algoritmo em instruções lógicas condicionais, que não exigem parenteses, e que apresentava um erro para o usuário. A Listagem 4.7 apresenta o trecho do código que permite o uso de expressões aritméticas em comandos que não requerem parênteses no seu formalismo, de acordo com a gramática dessas instruções para o MACP.

```
private String expArit() {
    String err = "";
```

```
if (conate) {
    if (posate) {
        posate = false;
    }
    op_relacional();
    conate = false;
}
```

Listing 4.7 – Trecho da correção para expressões condicionais aritméticas

A justificativa do trecho apresentado na Listagem 4.7 é que na rotina de verificação da construção de expressões aritméticas era definido para que fosse esperado, na regra até então de formação para a gramática, a obrigatoriedade de parênteses ou a expressão não era avaliada, restando como próximo passo apenas a busca por um identificador, em outros termos, uma variável. Isso resultava em vários erros ocasionados por elementos tidos como não esperados ou inválidos para a sentença, pois são termos que não tinham uma forma codificada no MACP que pudessem estabelecer um caminho na árvore geração de sua gramática.

Assim como nas situações anteriores, os usuários tinham alguns problemas durante a codificação de seus programas, pois baseados nos exemplos que eram apresentados como feitos pela aplicação e tidos como referências em sua documentação até então, deixava-os confusos, pois percebiam, de primeiro momento, como um erro na sua codificação que desenvolvera como solução. Dessa maneria gerava dúvidas em relação à interpretação das teorias passadas, Assim não conseguiam simular como era descrito e esperado como resposta para determinado problema.

4.4.3 Laço de Repetição para

O comando “para“, sendo uma instrução muito empregada como exemplo de repetição, e bastante adaptável para diversas situações, está presente na gramática do Portugol de todas as ferramentas apresentadas nesse trabalho. Então após os relatos de problemas pelos usuários e recomendações feitas para melhorar seu uso é que foram realizados algumas alterações que corrigissem as falhas relatadas e trouxessem as recomendações indicadas.

A anomalia descrita sobre as expressões aritméticas na Seção 4.4.2 afetava o comando “para“, no tocante das definições de inicialização e controle de término do laço, que ficava restrito a receber valores especificados por variáveis ou constantes numéricas.

Um problema relatado e corrigido foi a impossibilidade de se construir um laço de repetição com o comando ”para”em que o seu escopo de interação trabalhasse com decrementos, ao passo que o laço inicializado com determinado valor só poderia sofrer incrementos sucessivos, limitando ou dificultando o seu emprego em rotinas que funcionam com controles baseados em decrementos sucessivos do valor inicial do laço. Dessa forma, a correção foi permitir que a instrução ”passo”, pertencente à regra de construção do comando ”para”, pudesse receber trabalhar com valores negativos. As Listagens 4.8 e 4.9 apresentam as modificações na estrutura

da regra responsável por reconhecer a nova forma de iteração, que é necessária para auxiliar a resolução de tarefas.

```
Token aux = tk;
while (aux != null && !aux.getRepr().equalsIgnoreCase("passo")) {
    aux = aux.getNext();
}
if (aux != null) {
    aux = aux.getNext();
    int cond = toInt(aux.getRepr());
    if (cond == -101) {//Teste de passo negativo
        posate = true;
    }
}
```

Listing 4.8 – Trecho do código que verifica a ocorrência do passo negativo

Enquanto na Listagem 4.8, a parte de código apresentada trata do mapeamento de qual tipo de operação será aplicada dentro do contexto de incremento ou decremento para o comando "passo", a Listagem 4.9 a seguir contém o trecho de código criado para efetuar a verificação da regra de formação dentro da gramática aplicada ao MACP para a expressão fornecida como argumento do comando "passo", após a constatação, que se trata de um argumento negativo e que deve funcionar de forma decrescente na execução do laço "para".

```
if (tk.getClas() == 62 && passonegativo) {
    codigoGerado += tk.getRepr() + " ";
    tk = tk.getNext();
    err = fator();
    passonegativo = false;
} else {
    err = termo();
}
```

Listing 4.9 – Trecho do código que verifica expressão para passo negativo

Um quesito sugerido para melhorar a instrução "para", é poder facultar a necessidade de especificar o comando "passo" durante a codificação para o caso em que seu argumento seja para incrementos com valor constante positivo, unitário e igual a 1 (um). Essa facultatividade foi aplicada com o trecho de código demonstrado na Listagem 4.10, que é o responsável por fazer essa distinção no momento do seu emprego.

```
if (tk.getClas() == 82) {
    tk = tk.getNext();
    String sinal = "";
    int teste = toInt(tk.getRepr());
```

```
    if (teste != -101 && teste >= Integer.MIN_VALUE && teste <=
       Integer.MAX_VALUE) {
                    sinal = "+";
    }
    codigoGerado += "; " + codigoAux + " = " + codigoAux + " " +
       sinal + " ";
    err = expArit();
    if (!"".equals(err)) {
        return err;
    }
} else {
    codigoGerado += "; " + codigoAux + "+" + "+";
}
```

Listing 4.10 – Trecho do código para efetuar a facultação do comando passo

Com a implementação dessa característica o usuário do MACP não mais necessitará informar o comado "passo" para a situação trivial especificada anteriormente, onde a evolução da repetitividade ocorre em adições para um valor de incremento igual a 1 (um) e assim a instrução esperada para essa situação é a palavra reservada "faca".

A correção para expressões lógicas booleanas também favoreceu o comando "para", pois quando em uma implementação de um código Portugol pelo usuário no MACP, o condicional de controle do laço for descrito com situações que envolva variáveis do tipo lógicas e palavras reservadas que representem contantes booleanas, será reconhecido pela gramática formal aplicada ao MACP. Com isso, a instrução de iteração "para" fica melhor para o usuário aplicar aos seus programas.

4.5 Adição do Comando caso

A adição do comando "caso" é um recurso auxiliar a codificação dos programadores, que em vez de criar vários condicionais "se" e "senao", poderá criar um comando que tenha uma condição que possa ser avaliada para casos definidos em uma única estrutura de comando.

A seguir, a Listagem 4.11 mostra a porção inicial do código fonte que foi acrescentado ao MACP para reconhecer o comando "caso" na lista de tokens retornados pela fase de análise léxica. Para ser aceito pela gramática, houve a inserção, na tabela de marcas da linguagem, dos vocábulos necessários para definir como palavras reservadas do formalismo aceito pelo MACP conforme observado na Figura 4.12.

```
if (tk.getClas() == 84) {
    ++cscase;
    tk = tk.getNext();
```

Listing 4.11 – Trecho do código apresenta o inicio do reconhecimento do comando caso

A seguir, a inicialização das palavras reservadas apresentada na Listagem 4.12 está dividida em duas listas, isso se deve ao fato de algumas palavras terem sidos acrescentadas depois da conclusão das rotinas que validam as regras de formação estabelecidas na gramática. Portanto, os programadores anteriores da ferramentas verificaram que a adição de novas palavras na primeira lista impactaria em muitas alterações no analisador sintático, levando assim a uma solução alternativa, que é fazer uso de uma segunda lista. Se preciso, facilita a inclusão de uma nova marca para a linguagem.

```
public int isPalavraReservada() {
    String palavrasReservadas[] = {"variaveis", "inteiro", "real",
        "logico",
        "cadeia", "caractere", "inicio", "fim", "ler", "escrever",
            "se", "entao", "senao", "enquanto", "faca", "repita",
           "ate", "vetor", "de"};

    String palavrasReservadas2[] = {"para", "passo", "modulo", "
       caso", "opcao", "interrompa","limpar", "sair", "pausa"};
    for (int i = 0; i < 9; i++) {
        if (repr.equals(palavrasReservadas2[i]))  {
            return i + 81;
        }
    }
    for (int i = 0; i < 19; i++) {
        if (repr.equals(palavrasReservadas[i])) {
            return i + 1;
        }
    }
    return -1;
}
```

Listing 4.12 – Trecho do código da lista de palavras reservadas da gramática do MACP

Após a checagem do token "caso", o algoritmo passa para a próxima verificação, onde é esperada uma expressão que defina o parâmetro que será avaliado no comando "caso". Esse parâmetro poderá assumir qualquer forma válida para elementos que variem de sentenças lógicas e aritméticas a variáveis declaradas.

A sequência do reconhecimento da estrutura é realizada com o teste de obtenção das situações que representam os desvios que serão realizados na execução do algoritmo. Se o valor especificado pelo parâmetro corresponder a alguma situação definida em um dos desvios. O início desse procedimento poder ser observado na Listagem 4.13.

Em cada condição do comando "caso" é possível construir estruturas complexas em que todos os mecanismo formais determinados pela gramática do MACP podem ser usados na definição do conteúdo lógico de cada bloco de desvio implementado pelos usuários.

Durante o processo de desenvolvimento do comando "caso" foram realizados diversos testes utilizando instruções aninhadas dos comandos que já existiam no MACP, incluindo o próprio comando "caso", garantindo a generalidade que é necessária a gramática.

```
if (tk.getClas() == 23) {
    codigoGerado += "{ ";
    tk = tk.getNext();
    codigoGerado += "\n";
    ++ident;
    identa();
    Token token = null;
    while (tk != null) {
        if (tk.getClas() == 21 || tk.getClas() == 30
                || (tk.getClas() == 29 && tk.isType_cte() == 2001)
                || tk.getClas() == 41 || tk.getClas() == 42) {
            codigoGerado += "case " + tk.getRepr() + " ";
            tk = tk.getNext();
```

Listing 4.13 – Trecho do código do inicio das definições dos desvios que compõem o comando caso

Outro termo presente na estrutura do comando "caso" é a palavra reservada "senao", marca da linguagem que não tem a obrigação de ser definida durante a programação, e com significância semelhante quando empregado a comando "se", em que se nenhuma correspondência for localizada então todo conteúdo de seu escopo será executado.

O evento de alterar a cor da fonte do editor texto, o syntax highlight [1], teve melhorias inseridas devido a algumas palavras reservadas não se destacarem após a digitação ou terem apenas uma parte com realce de cor.

4.6 Tradução do código Portugol para outra linguagem

A tradução do algoritmo escrito em Portugol em linguagens que possuem perspectivas profissionais promove para o usuário a verificação da equivalência das instruções, introduzindo novas percepções da linguagem para a qual houve a transcrição. Isso permite que o usuário possa absorver gradativamente o significado dos vocábulos gerados pela tradução, reduzindo atritos decorrentes de uma mudança abrupta, priorizando a absorção de novos conhecimentos.

A exportação do código fonte escrito em Portugol para o C++, é um bom recurso para os usuários que querem expandir seus conhecimentos e assimilar uma nova linguagem de programação. No entanto, essa tradução é limitada no que diz respeito a nomes de identificadores. É que existe a possibilidade, por exemplo, da existência da declaração de uma variável em Portugol com o nome de "cin". Nesse caso, ao traduzir para C++ e tentar compilar

[1] Destaque de sintaxe que auxiliar distinção dos termos em um código fonte

o código gerado, o compilador pode projetar avisos e erros, visto que "cin"é uma função da biblioteca padrão, usada no código C++ resultante.

Da mesma maneira outra limitação está vinculada, a identificadores com caracteres underlines [2], também chamados de underscore, antes ou depois nos seus nomes, como "EXIT_SUCCESS", devem ser evitados, pois muitas constantes e identificadores internos utilizam essas padrões de nomeação, e seu uso pode implicar em conflitos durante a compilação.

4.7 Impressões durante o período de colaborador

O período de colaborador foi bastante construtivo, devido a expectativa que existia em torno da incumbência de poder agregar valor com soluções que atendessem as necessidades dos alunos, baseadas nos relatos e impressões que foram fornecidos durante o andamento da disciplina de Algoritmos I. Essa disciplina é justamente o estágio inicial de introdução a programação no curso de Ciência da Computação, e que adotou como ferramenta de auxílio ao aprendizado o MACP.

As contribuições inseridas ao MACP proporcionaram um grande aprendizado, que devido as possibilidades de contextualização de conhecimentos e experiências práticas aliados ao paradigma de programação colaborativa de um software de código aberto, ascendeu com uma maior veemência a importância e os benefícios que estão em projetos que disponibilizam o seu código fonte, pois permitem que colaboradores que se interessem por projetos semelhantes ao MACP possam trazer inovações, que permitam melhorar e ampliar as características dessas ferramentas. Outro ganho que é propiciado dessa política de desenvolvimento, é fazer com que as ferramentas possuam um suporte longo, à medida que desenvolvedores podem ir e vir para contribuir, adaptando-as às novas tecnologias prologando assim o ciclo de vida dessas ferramentas.

Boas experiências foram adquiridas das interações realizadas com os usuários, que resultaram em uma valorização de se manter um canal de comunicação ativo com os usuários, pois visto que nenhum software é perfeito e que por mais que se tenha cautela na codificação e nos testes, sempre há a possibilidade de ocorrerem erros não previstos, principalmente quando a complexidade de desenvolvimento da ferramenta cresce.

Inserir o aluno ao rigor de uma linguagem de programação, dita profissional e com uso de expressões em inglês, logo nas primeiras fases do estudo de programação, pode ser um entrave. É bastante comum que a maioria dos alunos, possuam uma bagagem do idioma inglês bem limitado, dificultando o entendimento e associação da teoria. Porém, ao adotar o Português Estruturado, ao aluno cabe apenas o esforço de compreender o formalismo adotado, relacionando os conceitos teóricos vinculados a abstração, que é um dos passos mais importantes na programação.

[2] São caracteres de sublinhado ou traço inferior

5 ANÁLISE E DISCUSSÃO DOS RESULTADOS

O presente capítulo traz alguns exemplos de programas escritos em Portugol aplicados ao MACP, demonstrando os resultados obtidos após as adições e correções aplicadas, corroborando as características discutidas ao longo deste trabalho.

A seguir, um exemplo de uso do comando "caso" na Listagem 5.1, que faz uso de uma constante lógica no controle do laço de iteração numa rotina elaborada para efetuar as operações matemáticas básicas entre dois números, informados pelo usuário. Nesse exemplo, é demostrado como está organizado às estrutura do comando "caso", esclarecendo quaisquer dúvidas quanto à maneira como deve ser usado, levando em consideração as etapas implementadas e discutidas na Seção 4.5.

```
variaveis
opt:inteiro;
a,b:real;
inicio
enquanto verdadeiro faca [[
  limpar;
  escrever '[1] Soma\n [2] Subtracao\n [3] Multiplicacao\n [4]
     Divisao\n [5] Sair\n';
  ler opt;
  caso opt [[
    1:
      ler a, b;
      a <- a + b;
      escrever 'soma = ', a, '\n';
    2:
      ler a, b;
      a <- a - b;
      escrever 'dif = ', a, '\n';
    3:
      ler a, b;
      a <- a * b;
      escrever 'mult = ', a, '\n';
    4:
      ler a, b;
      a <- a / b;
      escrever 'div = ', a, '\n';
    5:
      sair;
    senao
```

```
      escrever 'opcao invalida\n';
  ]];
  pausa;
]];
fim.
```

Listing 5.1 – Trecho do código de exemplo do comando caso

Após a correção do reconhecimento das constantes lógicas, testes foram efetuados para validar o trabalho realizado. Um exemplo é o apresentado pela Listagem 5.2, que possui um vetor do tipo lógico que receber atribuições sucessivas com base no valor lógico armazenado em uma variável.

A palavra reservada "logico" é um tipo de dado primitivo que possui dois valores, que podem ser falso ou verdadeiro. E que falso é diferente semanticamente de 0, vazio, nulo e para verdadeiro não há valores correspondentes a não ser o próprio valor verdadeiro.

A respeito do tipo lógico que faz parte da gramática do MACP, tem uma significância distinta em relação ao tipo inteiro, como na linguagem Java com o "true" ou "false", pois em algumas linguagens de programação como C e PHP, um valor ou expressão do tipo inteiro e diferente de zero é avaliado com verdadeiro nas instruções condicionais, levando a erros de lógica inesperados ocasionados por esse detalhe. Essa nuance não existe na gramática do MACP cujo benefício de que o tipo inteiro não terá a capacidade validar uma condição apenas pelo seu valor, assim necessitando que seja criada uma equação lógica que avalie o valor inteiro.

```
variaveis
l: logico;
v: vetor[1..10] de logico;
i: inteiro;
inicio
i <- 1;
l <- verdadeiro;
enquanto i <= 10 faca[[
  v[i] <- l;
  se l entao[[
    l <- falso;
  ]]senao[[
    l <- verdadeiro;
  ]];
  i <- i + 1;
]];
escrever '\n';
i <- 1;
enquanto i <= 10 faca[[
  escrever v[i], ' ';
```

```
  i <- i + 1;
]];
escrever '\n';
fim.
```

Listing 5.2 – Trecho do código de exemplo do uso de constantes lógicas

Uma programação efetuada ao MACP que requer atenção e detectada após a realização das alterações planejas durante o período de atividade como colaborador, e a sobrescrita de alguns dos eventos padrões do componente de texto. Então, como a detecção dessa codificação foi feita algum tempo depois, a análise desse trecho não foi realizada.

Um detalhe que pode ser observado é que os termos corrigidos ou adicionados como palavras reservadas mantiveram o padrão de colorização das palavras que compõem os tokens reservados a gramática do Portugol aplicado ao MACP.

O exemplo de código da Listagem 5.3 implementa um algoritmo de ordenação em que há a inicialização de vetor contido em um laço de repetição "para" que define explicitamente na sua estrutura o comando "passo". Ele tem definidos incrementos unitários de 1 (um) em 1 (um) a cada iteração. Ao passo que na implementação do algoritmo de ordenação, ambos os comandos "para" estão com a instrução "passo" facultada, e há o uso de uma variável, que é reconhecida com um identificador no avaliador de formação para expressões aritméticas, inicializado o laço mais interno, apresentando assim duas modificações comentadas anteriormente na Seção 4.4.3.

```
variaveis
v: vetor[1..10] de inteiro;
i, j, menor, tmp:inteiro;
inicio
para i de 1 ate 10 passo 1 faca[[
   ler v[i];
]];
para i de 1 ate 10 faca[[
   menor <- i;
   para j de i ate 10 faca[[
        se v[menor] > v[j] entao[[
          menor <- j;
        ]];
   ]];
   se menor <> j entao[[
      tmp <- v[menor];
       v[menor] <- v[i];
       v[i] <- tmp;
   ]];
]];
escrever '\n';
```

```
para i de 10 ate 1 passo -1 faca[[
    escrever v[i],' ';
]];
  escrever '\n';
fim.
```

Listing 5.3 – Trecho do código de exemplo do comando para

Uma outra observação que consta no exemplo da Listagem 5.3 é o emprego de um comando "passo" com valor de iteração negativo, atestando outra melhoria aplicada a estrutura da rotina de repetição "para". E de maneira semelhante ao token "caso", a instrução "passo" também recebeu uma cor de fonte que o identifica-se com uma palavra reservada.

A partir desses exemplos, que foram criados para atestar o funcionamento das programações efetuadas para o MACP tem-se a evidência do aumento do poder de expressividade da gramática. E assim facilitar e promover uma maior liberdade e flexibilidade para o usuário no momento de transformar suas ideias em programas, durante a fase de estudo.

Grande parte da detecção e recuperação de erros de um compilador tem origem na fase de análise sintática, pois através do fluxo de tokens reconhecidos pelo analisador léxico é que os erros que desobedecem as regras gramaticais que definem a linguagem são identificados (AHO; SETHI; ULLMAN, 1995).

E no MACP o seu Parser [1] é codificado para exibir mensagens em português dos erros mapeados com informações sobre linha e coluna, facilitando a localização e o entendimento do que causa o problema. Agregando para o usuário a importância da grafia correta das estruturas formais da gramática que compõem o código fonte.

Pesquisas sobre concepções teóricas de aprendizado enfatizam três mecanismos que definem como se estabelecem as relações de aquisição de conhecimento. Empirismo é quando os alunos são considerados apenas receptores de instruções. Racionalismo é o conhecimento adquirido do resultado de estruturas orgânicas inatas. Interacionismo é o conhecimento obtido mediante a contribuição tanto do sujeito quanto dos objetos do conhecimento (OLIVEIRA; COSTA; MOREIRA, 2001).

A partir do entendimento que circunda os três mecanismos que teorizam as relações de aprendizado, o ideal é que o alunos possam desenvolver a essência dessas características, de forma que possa mesclar a intersecção desses mecanismos. Para que seu engajamento com o aprendizado seja o mais simples, transformando as informações e experiências acumuladas pela interação do conteúdo teórico e pela prática em conhecimento.

O uso de ferramentas de cunho educativo permite a continuidade e um maior aprofundamento, uma vez que está pode acompanhar o aluno em ambientes que não seja o da instituição do curso, facilitando o acesso ao conteúdo.

Em relação a aparência, constata-se que toda a aplicação está suficientemente agradável. Porém, devido as contantes inovações dos recursos de interface promovidos pela evolução

[1] Termo em inglês que significa análise sintática

das linguagens de programação e pela criação de protótipos que aceleram o desenvolvimento baseado em métodos ágeis, há uma necessidade de se fazer atualizações gráficas para alimentar a satisfação visual do usuário.

De acordo com opiniões e pesquisas em usabilidade afim de definir uma quantidade mínima necessária para que testes possam fornecer resultados confiáveis. É que autores relatam que 3 participantes são necessários para descobrir 65% dos problemas, e que 5 participantes já fornecem 80% e que 9 participantes já podem contribuir com descoberta de 95% dos problemas de usabilidade das aplicações (VIRZI, 1992). A corroboração realizada por (NIELSEN, 1993) sobre estes estudos , relata que o aumento do número de participantes para além de cinco raramente leva a novas descobertas. Com base nesses autores, pode-se concluir que a turma de alunos, que eram superiores a 10, contribuíram para identificar boa parte dos problemas que existiam tornando a ferramenta mais estável.

A adição dessas modificações visuais impactam diretamente na usabilidade, pois o conforto que as novas técnicas de interação fornecem facilitam a interação do usuário com a interface. A experiencia computacional do usuário está intimamente ligada com a dificuldade que este possa vir ter ao interagir com interfaces antigas, ocasionada pela bagagem interativa que está vinculada a quais mecanismos de interação estão vigentes em um determinado contexto de tempo.

6 CONCLUSÃO

O que determina o software e suas possibilidades são as concepções que os usuários estimam no momento em que o usam, pois são essas percepções que definem o quanto um produto é capaz de oferecer facilidades que melhorem a experiência do usuário em realizar suas tarefas. Isso fomenta conhecimentos amparados pelo uso das tecnologias, estendendo o ambiente de aprendizado e ampliando os canais de comunicação. E assim desenvolvendo uma formação volta para melhor adaptação dentro de uma sociedade que a cada dia está mais conectada independente da área de formação e atuação.

As questões abordadas sobre a adoção de ferramentas que auxiliem o aprendizado das teorias de programação, fazem o apelo aos professores, que sempre que for possível e levando em consideração as características dos alunos e das metodologias empregadas, utilizar-se de ferramentas que amplie o suporte para o aluno durante a fase de aprendizado.

A adoção de uma ferramenta não implica em garantias de melhoria de rendimentos dos alunos, e tão pouco é uma solução para eliminar o número geralmente elevado de reprovações e abandonos. pois não constitui, e não há uma fórmula pronta que resolva e preveja todos os problemas inerente ao processo de ensino e aprendizado.

O presente trabalho traz informações sobre características que fazem parte de um consenso entre diversos autores que discursam sobre o tema da aplicação de software na educação, enfatizando os benefícios que estão atrelados a essa prática do ponto de vista do aluno e do professor. Sendo o MACP uma ferramenta que evoluiu de um trabalho de cunho curricular acadêmico para um projeto que possa atender esses objetivos, é que se faz necessária ter uma continua avaliação e adaptação dos recursos para torna-lo mais eficaz.

Todas as alterações implementas ao MACP estão engajadas em levar condições que facilitem o processo de codificação dos usuários fornecendo recursos que maximizem a experiência do aprender, trazendo a flexibilidade na decisão de como será resolvido determinado problema de programação que possa ser promovido durante o período de aprendizado.

O resultado adquirido na participação de um projeto com o MACP é a possibilidade de ter a experiência e engajamentos normalmente recebidos nos ambientes mais profissionais, e que foi experimentado ainda durante o período de graduação agregando uma percepção melhor lapidada sobre o processo de construir programas. Tendo ainda os relatos fornecidos pelos usuários para melhor direcionar o foco dos esforços de desenvolvimento.

Finalizando, a adoção de disponibilizar o código fonte do MACP sob a licença de livre cópia e modificação dá uma maior chance para que, mesmo que ocorra dos mantenedores oficiais deixarem de dá suporte a ferramenta, outros entusiastas com a ferramenta e com o tema de se desenvolver uma ferramenta de auxílio a programação para cursos que não necessariamente sejam especificamente relativos a computação.

6.1 Trabalhos futuros

Uma característica bastante válida para modificações futuras e que está presente em todas as ferramentas que foram relatadas neste trabalho junto com o MACP, é existência de um mecanismo de acompanhamento de execução. Isso auxilia o usuário no acompanhamento passo a passo do seu programa facilitando a identificação de erros e o entendimento da organização do seu código fonte.

Uma funcionalidade que seja capaz de demonstrar de forma gráfica e intuitiva a construção de um programa é também um recurso bastante útil, pois facilita a visão geral do fluxo de execução. Essa foi uma característica bem elaborada pela ferramenta Portugol Studio que permite a tradução de um código fonte em Portugol para uma visão gráfica e uma implementação gráfica para código Portugol.

Todas as modificações acima mencionadas para serem adicionadas necessitam que todo o projeto do MACP seja refatorado e dimensionado, para que possa facilitar a manutenção do código fonte da ferramenta, e que novos recursos possam ser agregados sem as mesmas dificuldades encontradas no momento em que as modificações descritas neste trabalho foram inseridas.

6.2 Considerações finais

O resultado de treinar os estudantes a exibir determinado comportamento usando métodos para reforçar o comportamento desejado, implica na busca de instrumentos que possam suprir a incapacidade do professor, sozinho, dar suporte a todos os alunos ao mesmo.

A proposta do MACP, como uma ferramenta de apoio ao aprendizado de programação, está bem definida e tem recursos suficientes para que os usuários e professores a adotem durante as primeiras fases dos cursos de ensino a programação.

Referências

AHO, A. V.; SETHI, R.; ULLMAN, J. D. **Compiladores Principios, Técnicas e Ferramentas**. [S.l.], 1995. Citado na página 29.

BEAUBOUEF, T.; MASON, J. **Why the high attrition rate for computer science students: some thoughts and observations**. [S.l.], 2005. In: ACM SIGCSE Bulletin, USA, v. 37, n. 2, p. 103-106, jun. Citado na página 8.

BUTLER, M.; MORGAN, M. **Learning challenges faced by novice programing students studying high level and low feedback concepts**. [S.l.], 2007. Citado na página 8.

CARROLL, J. M. **Designing Interaction: Psychology at the human computer interface**. [S.l.], 1991. Disponível em: <https://www.cs.colorado.edu/~martin/Csci6402/Papers/carroll97.pdf>. Citado na página 7.

GIANGRANDE, J. E. **CS1 Programming Language Options, Journal of Computing Sciences in Colleges**. [S.l.], 2007. Citado na página 8.

GOMES, A.; MENDES, A. **Suporte à aprendizagem da programação com o ambiente SICAS**. [S.l.], 2000. Citado na página 8.

GURGEL, I. **A Importância de Avaliar a Usabilidade dos Jogos: A Experiência do Virtual Team. In: SBGAMES - Anais do Simpósio Brasileiro de Jogos de Computador e Entretenimento Digital**. Recife-PE, 2006. Disponível em: <http://www.sbgames.org/papers/sbgames09/computing/short/cts30_09.pdf>. Citado na página 4.

MANSO, A.; OLIVEIRA, L.; MARQUES, C. G. **Ambiente de Aprendizagem de Algoritmos – Portugol IDE**. [S.l.], 2009. Disponível em: <'http://orion.ipt.pt/~manso/papers/2009/Portugol%20Challenges2009.pdf'>. Citado na página 6.

MERCADO, L. P. L. **Novas tecnologias na educação: Reflexões sobre a prática. Maceió: EDUFAL**. [S.l.], 2002. Citado na página 11.

MORENO, A.; MYLLER, N. **Producing an Educationally Effective and Usable Tool for Learning, The Case of Jeliot Family**. [S.l.], 2003. Disponível em: <https://www.cs.colorado.edu/~martin/Csci6402/Papers/carroll97.pdf>. Citado na página 6.

NIELSEN, J. **Usability Engineering**. Boston - USA, 1993. Citado na página 30.

OLIVEIRA, C. C.; COSTA, J. W. da; MOREIRA, M. **Ambientes informatizados de aprendizagem: Produção e avaliação de software educativo**. Campinas-SP, 2001. Disponível em: <http://www.multimeios.ufc.br/arquivos/pc/monografias/Monografia_kerley.pdf>. Citado 2 vezes nas páginas 14 e 29.

PATTIS, R. E.; ROBERTS, J.; STEHLIK, M. **Karel the Robot: A Gentle Introduction to The Art of Programming**. [S.l.], 1995. Citado na página 14.

QUINTELA, B. de M.; RIBAS, A. M. **ENSINAR E APRENDER ALGORITMOS E PROGRAMAÇÃO NO ENSINO SUPERIOR: Desafios e melhores práticas**. [S.l.], 2016. Disponível em: <https://www.researchgate.net/publication/310628856_ENSINAR_E_APRENDER_ALGORITMOS_E_PROGRAMACAO_NO_ENSINO_SUPERIOR_Desafios_e_melhores_praticas>. Citado na página 15.

SOUZA, C. M. de. **VisuAlg**. [S.l.], 2016. Disponível em: <http://www.apoioinformatica.inf.br/produtos/visualg/>. Citado na página 5.

SOUZA, M. B. de. **Uma Abordagem Metodológica voltada para o Ensino-Aprendizagem de Algoritmos**. [S.l.], 2013. Citado na página 12.

TEIXEIRA, A. C.; BRANDAO, E. J. R. **Democratização do conhecimento: repensando o processo de exclusão social. Revista 9 Novas Tecnologias na Educação. CINTED-UFRGS, V. 1 ,nº 1, Fevereiro**. [S.l.], 2003. Citado na página 8.

VALENTE, J. A. **Computadores e Conhecimento: repensando a educação**. [S.l.], 1993. Citado na página 10.

VIEIRA, F. M. S. **A Utilização das Novas Tecnologias na Educação numa Perspectiva Construtivista**. [S.l.], 2005. Citado na página 10.

VIRZI, R. A. **Refining the test phase of usability evaluation: how many subjects is enough? In: Human Factors**. [S.l.], 1992. Citado na página 30.

WIKIPEDIA. **Portabilidade**. [S.l.], 2017. Disponível em: <https://pt.wikipedia.org/wiki/ISO/IEC_9126>. Citado na página 7.

ZANINI, A. S.; RAABE, A. L. A. **Análise dos enunciados utilizados nos problemas de programação introdutória em cursos de Ciência da Computação no Brasil**. [S.l.], 2012. In Anais do XXXII Congresso da Sociedade Brasileira de Computação, XX WEI – Workshop sobre Educação em Computação, Curitiba. Citado na página 10.