II Encontro de Ciência e Tecnologia do IFSUL Campus Bagé

DESCOMPLICANDO A PROGRAMAÇÃO EM LINGUAGEM C. UMA SOLUÇÃO PARA DEPURAÇÃO SIMPLES DE CÓDIGOS.

GOMES, M. S. [1], AMARAL, E. M H. [1]

[1] Universidade Federal do Pampa (UNIPAMPA) – Bagé – RS – Brasil

RESUMO

Este trabalho apresenta um projeto que visa construir uma ferramenta para auxiliar a depuração de códigos em linguagem C, por alunos iniciantes. Para alcançar este objetivo foram realizadas analises dos principais erros cometidos pelos alunos durante aulas práticas de programação com intuito de servir de base para a implementação deste recurso. Esta ferramenta está em desenvolvimento e já é possível apresentar como será o seu funcionamento e em que pode auxiliar o aluno em sua aprendizagem na disciplina de algoritmos e programação ou similar.

Palavras-chave: Algoritmos e Programação; Dificuldades na Prática de Programação; Depuração de Códigos.

1 INTRODUÇÂO

As dificuldades inerentes ao processo de ensino e aprendizagem de algoritmos e programação estão relacionadas a diversos fatores e com intuito e amenizar as dificuldades enfrentadas pelos acadêmicos, inúmeros grupos trabalham na realização de experimentos práticos e na construção de ferramentas e metodologias para facilitar o processo de ensino e aprendizagem dessa área, como: Yang *et al.* (2014) e Likke *et al.* (2014).

Para More *et al.* (2011), a correção dos erros é uma das tarefas mais importantes na programação, chamada de depuração de códigos, sendo uma das causas de alunos obterem baixo desempenho nesta disciplina. Os estudantes enfrentam dificuldades em representar suas abstrações em comandos para linguagem de programação, assim como entender e interpretar as mensagens de erros exibidas pelo compilador e por consequência corrigir os erros de sintaxe.

Do ponto de vista construcionista, Papert (1986), cometer um erro não significa algo condenável, muito pelo contrário: o erro é tido como oportunidade ideal para a construção do conhecimento. Almeida (1999) destaca que o erro passa a ser um revisor de ideias e não mais um objeto de frustração. Valente (1999) também diz que o processo de achar e corrigir um erro constitui uma oportunidade única para o aprendiz aprender sobre um determinado conceito envolvido na solução do problema ou sobre estratégias de resolução de problemas.

Nota-se, portanto, que após o aluno receber o *feedback* do computador sobre a execução de seu programa, ele passa a tentar identificar a origem de seu erro e saná-lo, ou ainda, empenhar-se na construção de melhorias em seu programa. Esse processo de depuração é, para o aluno, um momento de pensar sobre o pensar [Valente 1999]. A atividade de depuração na programação é muito importante do ponto de vista da aprendizagem e é exatamente por isso que deve ser estimulada.

Tomando como base teorias e falas de autores aqui citados, este trabalho visa desenvolver uma ferramenta que torne mais fácil a depuração de códigos em linguagem C, que é o padrão utilizado na instituição, tornando a apresentação de mensagens de erro para o usuário mais simples e na linguagem materna. E, desta forma estimular a aprendizagem de programação por parte dos alunos.

2 METODOLOGIA (MATERIAIS E MÉTODOS)

A pesquisa realizada neste trabalho foi classificada quanto a sua natureza como aplicada e, exploratória quanto aos objetivos. A metodologia adotada para o desenvolvimento deste trabalho consistiu inicialmente no levantamento de um referencial bibliográfico. A etapa seguinte, denominada passo 2, foi desenvolvido um instrumento de coleta de dados para esta pesquisa, onde foi aplicado a uma turma de algoritmos e programação. Este instrumento foi utilizado durante aproximadamente 3 meses. Em seguida, passo 3, os dados foram analisados para identificar os erros mais comuns cometidos pelos alunos na prática de programação em linguagem C.

No passo 4 foi dado início a implementação da ferramenta proposta neste trabalho. No passo 5 a ferramenta desenvolvida será testada pelos alunos para que no passo 6 se possa comparar os resultados obtidos com a sua aplicação e poder validar a sua proposta.

3 IMPLEMENTAÇÃO

Nesta seção, serão apresentados o desenvolvimento, aplicação e os resultados conseguidos com o instrumento de coleta de dados e, será também mostrado o desenvolvimento da ferramenta proposta.

3.1 INSTRUMENTO DE COLETA DE DADOS

Com a finalidade de coletar informações sobre os erros cometidos por alunos iniciantes em programação na linguagem C, foi realizado a análise e desenvolvimento de uma ferramenta de monitoramento das saídas geradas durante o processo de compilação utilizando o compilador GCC.

Para a implementação da aplicação, denominada CFacil que será também o nome da ferramenta desenvolvida, utilizou-se a tecnologia de Shell Script. A arquitetura de funcionamento do software é da seguinte maneira: ocorre a solicitação ao usuário de um nome para o arquivo de saída e o nome do arquivo que se deseja compilar. Após isto, o arquivo enviado pelo usuário é compilado duas vezes, uma para ser apresentado no terminal e outra para ser enviado para o arquivo que contém os erros.

A coleta de dados, com a aplicação, ocorreu em uma turma da disciplina de algoritmos e programação, com um universo amostral de 22 alunos por aula, todas realizadas em um laboratório de programação. As atividades acadêmicas da disciplina ocorreram em encontros semanais, 4 horas/aulas, em um período total de aproximadamente 3 meses. Todas as atividades de programação seguiram o plano de ensino, sendo que a utilização do CFacil ocorreu com supervisão docente, onde os estudantes realizavam as tarefas de programação propostas, com o armazenamento dos logs de compilação.

Com o objetivo de destacar os erros com maior incidência durante o experimento, adotou-se o Sobek1, um software de mineração de dados, utilizado para identificar conceitos/termos mais relevantes em um determinado texto, a partir da análise de frequência destes no material textual [Klemann 2011]. A partir da aplicação deste no arquivo logs.txt foi possível gerar um diagrama com a frequência dos principais termos e suas relações, conforme a Figura 06, onde percebesse que os termos mais frequentes nesta análise são *error* e *warning*, além do termo *function* que sempre está presente nas compilações, pois indica em qual função encontra-se o erro/*warning*. Destaca-se também o termo *error expected*, onde ocorrem os erros nas funções *printf*, *scanf* e a falta de *tokens*. Os gráficos 01 e 02 apresentam os resultados conseguidos com a aplicação do Sobek.

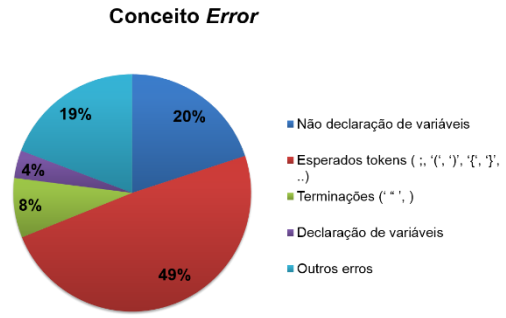

Gráfico 1. Resultados observados para o conceito *error*

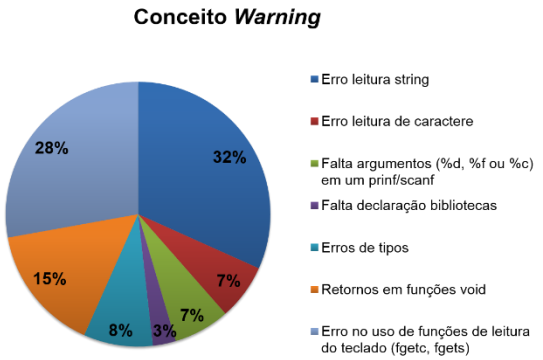

Gráfico 2. Resultados observados para o conceito *warning*

3.2 DESENVOLVIMENTO DA APLICAÇÂO

A partir dos resultados conseguidos com a coleta dos dados foi dado início a implementação da aplicação. Utiliza-se para a codificação da mesma a própria linguagem C, pois a avaliação neste momento não será realizada com base em uma interface, mas sim somente os *logs* de compilação são de interesse para poder avaliar a efetividade da ferramenta.

O software a ser desenvolvido realiza a análise de cada uma das estruturas da linguagem C, como: *if-else*, *while*, *do-while*, *for*, *switch*, comandos de entrada e saída, declaração de bibliotecas, declaração de variáveis a procura de erros que tenham sido cometidos. A implementação partiu do princípio de que todos os programas C consistem de uma coleção de uma ou mais funções, mais variáveis globais. Uma função começa por um tipo, seguida de um nome, parâmetros, e um bloco de código associado a ela. Um bloco começa com '*{*' seguido de comandos e termina em '*}*'. Um comando é uma palavra-chave, como *if* seguida de um bloco de código, ou uma expressão aritmética ou condicional.

Primeiramente o arquivo a ser compilado pelo usuário é analisado e identificado todas as funções com o seu respectivo tipo e parâmetros, e guardada a sua localização no arquivo, além disto são identificadas também todas as variáveis globais. Após isto, é procurado no código a função *main,* ou seja a função principal que deve sempre estar presente em programas C, caso isso não ocorra é informado o erro para o usuário.

[1] Disponível em: http://sobek.ufrgs.br/index.htm

A partir da identificação da função *main* é dado início a validação do bloco de código, que pode conter as estruturas *if*, *for*, *while*, expressões entre outras, que são tratadas individualmente.

4 RESULTADOS E DISCUSSÃO

Apesar da implementação da ferramenta não estar completa já é possível ter alguns resultados a partir de testes realizados pelo desenvolvedor, já que ainda não foi testada pelos alunos. A montagem na figura a seguir, Figura 01, apresenta alguns erros reconhecidos pela aplicação.

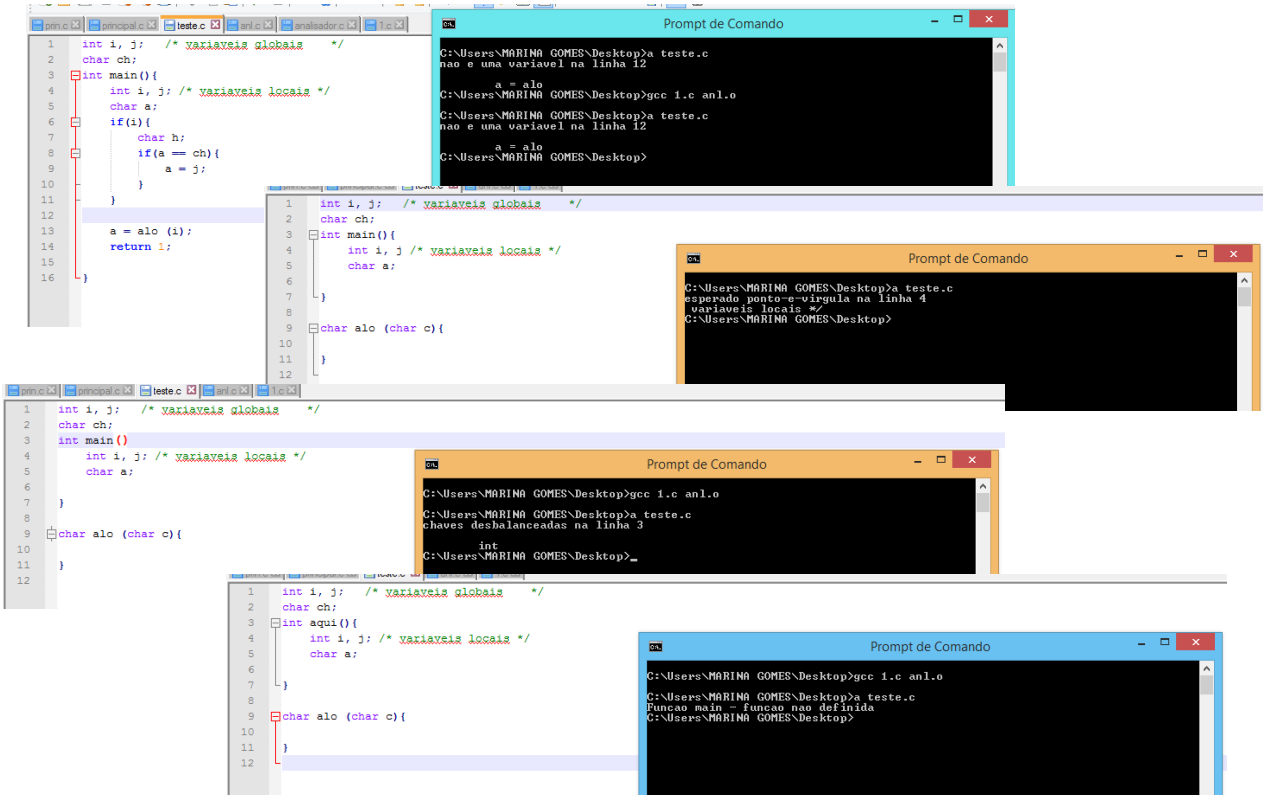

Figura 1. Erros reconhecidos pela aplicação

5 CONCLUSÕES

A partir dos resultados observados com os testes durante o desenvolvimento da aplicação, é possível inferir que a ferramenta tem potencial para servir de auxílio para o aluno durante as práticas de programação, e também para o professor que poderá utilizar este tempo que era necessária para correção de erros de compilação no atendimento aos alunos, para focar a sua atenção na maneira como o aluno está evoluindo na disciplina e quais são os pontos que necessitam de um melhor detalhamento e atenção.

6 REFERÊNCIAS

Almeida, M. E. B., (1999). A Informática Educativa na Usina Ciência da UFAL. Anais do II Seninfe, NIES/UFAL, Maceió.

Klemann, M., Reategui, E., Rapkiewicz, C. (2011). Análise de Ferramentas de Mineração de Textos para Apoio à Produção Textual. Anais do XXII Simpósio Brasileiro de Informática na Educação.

Likke, M., Coto, M., Mora, S., Vandel, N., Jantzen, C. (2014). Motivating programming students by Problem Based Learning and LEGO robots. In: Global Engineering Education Conference (EDUCON, IEEE).

More, A., Kumar, J., Renumol, V. (2011). Web Based Programming Assistance Tool for Novices. In: IEEE International Conference on Technology for Education.

Papert, S. (1986). Construcionism: a new opportunity for elementar Science education. Cambridge, Epistemology and Learning Group, Massachusetts Institute of Technology.

Valente, J. (1999). Análise dos diferentes tipos de softwares usados na Educação, In: Valente, J.A. (org). O computador na sociedade do conhecimento. Ed. Campinas: UNICAMP/NIED.

Yang, T., Yang, S., Hwang, G. (2014). Development of na Interactive Test System for Students Improving Learning Outcomes in a Computer Programming Course. In: Advanced Learning Technologies (ICALT).