Estudo de caso sobre o uso combinado de Técnicas de Análise Estática para identificação de plágio de *software* em códigos-fontes

Jose Luis A. Nunes[1], Fernando Sambinelli[2]

[1]Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas – Instituto Federal de Educação, Ciência e Tecnologia de São Paulo (IFSP) Campus Hortolândia – SP - Brasil

[2]Área de Informática – Campus Hortolândia – Instituto Federal de Educação, Ciência e Tecnologia de São Paulo

joselanunes@gmail.com, sambinelli@ifsp.edu.br

Abstract. *Plagiarism is a chronic problem, in addition to being a bad practice, it injures copyright, although Brazilian Law No. 9.609/98 helps protect the software producer's intellectual property, it is necessary to identify whether or not no evidence of plagiarism, in this sense, this article investigates the combined use of static analysis techniques to identify evidence of software plagiarism in source codes in a restricted and academic context. The techniques of syntactic similarity and semantic similarity were applied and the results obtained proved to be effective in infer whether there is evidence of software plagiarism.*

Resumo. *O plágio é um problema crônico que, além de ser uma prática malvista, fere os direitos autorais. Apesar de a Lei Brasileira n.º 9.609/98 ajudar a proteger a propriedade intelectual do produtor de software, faz-se necessário prover métodos e ferramentas que auxiliem na identificação de indícios de plágio. Este trabalho tem por objetivo analisar o uso combinado de técnicas de análise estática de código, para identificação de indícios de plágio de software num estudo de caso no contexto acadêmico. Foram aplicadas as técnicas de similaridade sintática e similaridade semântica, e os resultados obtidos mostram-se efetivos para inferir com mais segurança sobre a possibilidade de haver plágio de software.*

1. Introdução

Dos primórdios da globalização até o início da pandemia do coronavírus, no final do ano de 2019, a rede mundial aumentou sua base de usuários exponencialmente ao longo desses anos. Um estudo promovido pelo *Hootsuite* em parceria com a agência *We Are Social* apontou que aproximadamente 4,66 bilhões de pessoas estão conectadas à *Internet* atualmente, número representado pela proporção de 6 em cada 10 pessoas que acessam a rede por meio de um computador, tablet ou smartphone.

> [...] 4,66 bilhões de pessoas em todo o mundo usam a Internet em janeiro de 2021, um aumento de 316 milhões (7,3%) desde então no ano passado. A penetração global da Internet agora é de 59,5%. No entanto, o COVID-19 teve um impacto significativo no

> relatório de números de usuários da Internet, portanto, os números reais podem ser maiores [We Are Social e HootSuite - Digital 2021].

Com estimados 7,83 bilhões de pessoas habitando o planeta, a base de usuários da *Internet* cresceu 7,6%, no intervalo de um ano, enquanto o número global de habitantes cresceu menos de 1% em sua soma, apontando uma rápida expansão da rede mundial ao redor do mundo. Uma vez que passamos cada vez mais horas nesse mundo digital, torna-se fundamental criar mecanismos para proteger os direitos autorais dos desenvolvedores de *software*, os quais são os criadores desse mundo digital.

> [...] a população mundial era de 7,83 bilhões no início de 2021. As Nações Unidas relatam que esse número está crescendo 1% ao ano, o que significa que o total global aumentou em mais de 80 milhões de pessoas desde o início de 2020 [We Are Social e HootSuite - Digital 2021].

Esse crescimento exponencial de conteúdo influi diretamente em ações que violam a propriedade intelectual, pois crimes relacionados a infrações de direitos autorais têm crescido cada vez mais, e desenvolver meios que auxiliem na prevenção e detecção do roubo de propriedade intelectual torna-se fundamental.

Segundo Silva e Domingues (2008), a expansão da *Internet*, com o acesso a diversos repositórios dos mais variados assuntos, aumentou consideravelmente a incidência de plágio, principalmente no meio acadêmico. Para Culwin e Lancaster (2000), o plágio é roubo de propriedade intelectual, em que se inclui a utilização de código de programa sem permissão ou referência como um dos métodos de plágio.

Em razão dos argumentos citados, este trabalho tem como foco a análise de técnicas e ferramentas que auxiliem a identificação de indícios de plágio de *software*. Esta identificação pode ser realizada por meio da análise estática dos artefatos do *software*, ou seja, são analisados os elementos referentes à linguagem de programação, modelagem de dados, arquitetura, código-fonte, bibliotecas externas, entre outras documentações.

Diante dessa temática, surgiram as questões: a identificação de plágio de *software* pode ser feita combinando mais de uma técnica de análise estática? O uso combinado de técnicas de análise estática para identificação de plágio é mais vantajoso que o uso isolado?

O objetivo geral deste trabalho é avaliar, por meio de um estudo de caso, os efeitos do uso combinado de técnicas de análise estática, particularmente, as técnicas de similaridade sintática e similaridade semântica, para auxiliar na identificação de indícios de plágio de *software* em cinco códigos-fontes de um jogo chamado Jogo da Memória que consiste em memorizar uma sequência de 16 números distribuídos em 4 colunas e 4 linhas, desenvolvidos na linguagem de programação C, elaborados por estudantes do curso de Análise e Desenvolvimento de *Software* do Instituto Federal de São Paulo, no campus Hortolândia.

2. Aspectos legais do *software* no Brasil

Independentemente de como ocorra o plágio de *software*, faz-se necessário o entendimento das pessoas de que essa prática é um ato criminoso e passível de severas penas, pois, de acordo com a Lei nº 9.609/98 (Lei do *Software*) e seu regulamento, Decreto nº 2.556/98, e a "Lei de Direito Autoral" nº 9.610/98, os programas de computador — seja em código-fonte ou objeto — devem ser protegidos, assim como as obras literárias. A pena para a violação de direitos autorais de programa de computador, conforme o artigo 12 da Lei nº 9.609/98, aponta que é plausível a detenção de seis meses a dois anos ou multa. Se a violação for para fins de comércio, a pena passa para reclusão de 1 a 4 anos e multa conforme o mesmo artigo.

> Art. 2º O regime de proteção à propriedade intelectual de programa de computador é o conferido às obras literárias pela legislação de direitos autorais e conexos vigentes no País, observado o disposto nesta Lei [LEI Nº 9.609, DE 19 DE FEVEREIRO DE 1998.].

De modo geral, existem três tipos de *software*:

- *Software* de domínio público: aquele cujos autores liberam o direito de propriedade intelectual dos seus produtos. Este tipo de *software* é comum no mundo acadêmico.

- *Software* licenciados ou comerciais: são os produtos comercializados por meio de licenças, de acordo com o capítulo IV, "Dos contratos de licença de uso, de comercialização e de transferência de tecnologia", que abrange dos artigos 9º ao 11 da Lei do *Software* (Lei nº 9.609/98).

- *Freeware*: são os *softwares* gratuitos, utilizados livremente, sem ter-se que pagar por isso. São comuns os que são gratuitos para pessoas físicas, mas com uma versão *shareware* para pessoas jurídicas.

O modo de garantir legalmente a propriedade intelectual de uma criação de *software* é relativamente simples, basta o criador do sistema registrá-lo no Instituto Nacional de Propriedade Industrial (INPI). O registro assegura e decreta os direitos do criador e sua exclusividade sobre o *software* criado e descrito no registro, esse reconhecimento é internacional e tem prazo de validade de 50 anos. No entanto, no Brasil, o *software* não precisa ser registrado para que esteja protegido pela Lei de Direito Autoral, podendo ser objeto de ações jurídicas de violação de direitos autorais. Quando ações de violações autorais chegam a julgamento, normalmente, um perito em informática é nomeado pelo juiz responsável, para comprovar se há indícios de plágio de *software* ou não. Neste caso, faz-se necessário ao perito a utilização de técnicas e ferramentas adequadas para comprovar indícios de plágio e auxiliar o juiz a tomar decisões corretas e baseadas na ciência.

3. Metodologia

A primeira etapa do projeto consiste na revisão bibliográfica, visando compreender os conhecimentos presentes na literatura sobre as técnicas e as ferramentas de análise estática para identificação de plágio de *software*.

Após o entendimento das técnicas e conceitos envolvidos, o foco passou a ser a coleta dos códigos-fontes dos programas elaborados pelos envolvidos no estudo de caso. Os nomes e dados dos alunos que participaram neste trabalho foram totalmente protegidos, sem exposição de nenhuma informação pessoal e identificável. Em posse dos códigos-fontes, é realizada a análise estática dos programas coletados, para isso são analisados os programas executáveis e seus respectivos códigos-fontes, sendo aplicadas as técnicas de análise de similaridade sintática e análise semântica, tais como comparação textual e *disassembler*.

A técnica disassembler é feita por meio da engenharia reversa, que recupera a representação simbólica das instruções de um arquivo binário, no nosso caso, de programas em linguagem máquina x86 controlada pelo Sistema Operacional Windows. Desta forma, é possível ver o funcionamento do *software* através de quadros sequenciados, chamamentos, linhas de códigos, sendo estes códigos compilados na linguagem de programação C [Monteiro, Gorayeb, Monteiro e Noleto 2018].

Para fazer a engenharia reversa dos dados obtidos, é utilizada a ferramenta IDA Pro *Disassembler* [IDA Pro], pela qual são analisadas as chamadas feitas para as bibliotecas de cada um dos programas, e assim é possível ver as suas similaridades. Também são analisadas as rotinas, pois, apesar de poderem ter nomes diferentes, faz-se necessário analisar as suas funções em detalhes [Monteiro, Gorayeb, Monteiro e Noleto 2018].

Após finalizada a etapa de análise, foram descritos os resultados sobre a identificação dos indícios de plágio no estudo de caso. Também serão relatadas as vantagens da utilização combinada de técnicas de identificação de plágio de *software*, a Figura 1 representa em forma de diagrama todas as etapas do envolvidos descritas acima e que compõem o estudo de caso.

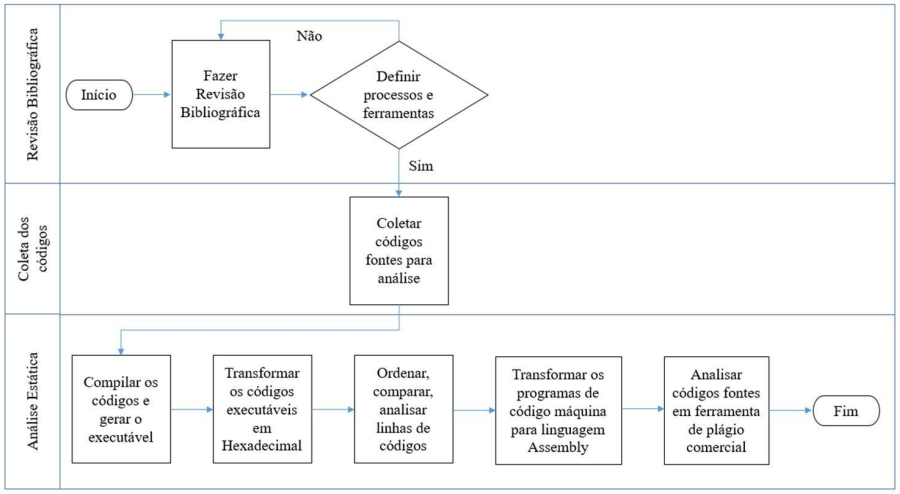


Figura 1. Visão geral das etapas do estudo de caso

4. Desenvolvimento

Nesta subseção, será apresentado o embasamento teórico estudado e aplicado no estudo de caso, considerando-se programas desenvolvidos na linguagem de programação C e as ferramentas necessárias.

4.1. Ambiente de desenvolvimento integrado

Para fazer este estudo de caso, foram usados cinco códigos-fontes de um programa chamado Jogo da Memória, que foi solicitado para ser desenvolvido por cinco grupos de alunos que cursaram a disciplina de Linguagem de Programação I, no Instituto Federal de Educação, Ciência e Tecnologia, de São Paulo, Campus de Hortolândia. Em paralelo, e com o intuito de validar a efetividade das técnicas de análise estática (similaridade sintática e semântica), foi feito o mesmo experimento em dois programas similares, que simulam uma Calculadora, nesse caso a única diferença entre esses dois programas é que um deles está todo escrito em português, ao passo que o outro está todo escrito em inglês.

Em ambos os experimentos, seja quanto aos cinco programas do Jogo da Memória ou quanto aos dois programas de Calculadora, a primeira etapa consistiu em coletar os códigos-fontes dos programas desenvolvidos e, com auxílio da ferramenta de ambiente de desenvolvimento integrado, chamada *Netbeans*, executar os programas. Vale ressaltar que todos os programas foram executados no mesmo computador e compilador, de forma que seja possível manter o mesmo padrão para análise, com isso é possível garantir que todos os programas estão compilando com sucesso, além disso, na própria ferramenta *Netbeans*, é possível identificar visualmente se os códigos são similares uns aos outros, mas ainda assim, essa análise por si só é insuficiente e carece de fundamentos mais sólidos para se inferir sobre um possível plágio. Após a compilação dos cinco programas do Jogo da Memória e dos dois programas de Calculadora, foram gerados os cinco arquivos binários para o Jogo da Memória e dois arquivos binários para o programa Calculadora. Os cinco programas do Jogo da Memória estão representados na ferramenta *Netbeans* como Grupo_1 até o Grupo_5, e os dois programas de Calculadora estão denominados como Exer_Funcao1 e Exer_Funcao2.

Na Figura 2 é apresentado o ambiente de desenvolvimento integrado denominado Netbeans onde os cinco programas do Jogo da Memória e os dois programas de Calculadora foram compilados.

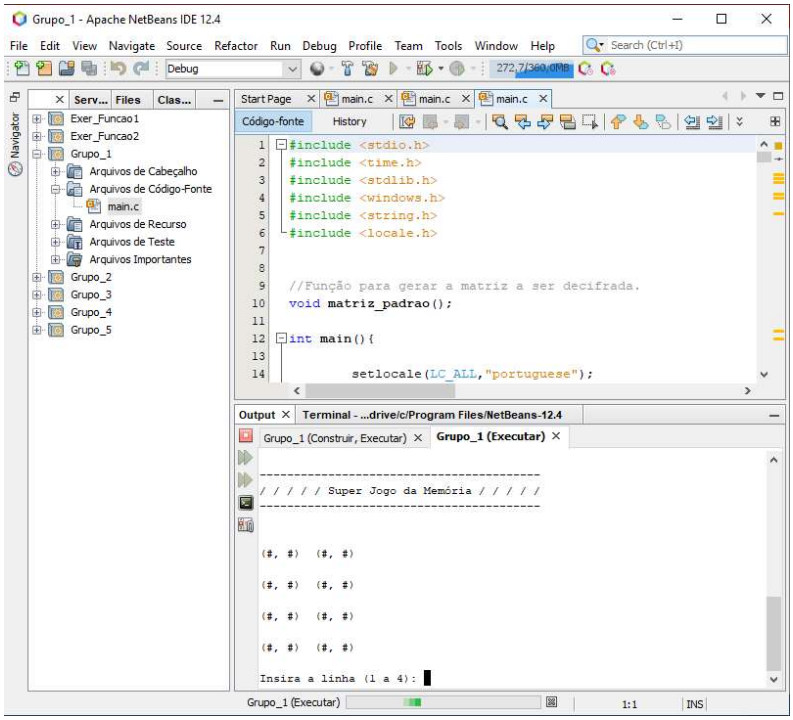


Figura 2. Exemplo dos programas na ferramenta *Netbeans*

4.2. Análise Estática – Comparação Textual (Hexadecimal)

Segundo Mota e Goya (2016), em sua maioria, as técnicas de detecção de plágio realizam a comparação de dois ou mais documentos, procurando analisar o grau de similaridade entre eles. De posse de uma medida de similaridade e de uma descrição da técnica usada para obter tal medida, um julgador poderá avaliar quem tem razão em um processo de litígio de propriedade intelectual.

Desta forma, a comparação textual nesse estudo de caso consiste na transformação dos códigos executáveis em linguagem C para hexadecimal. Para isso, o primeiro passo consistiu na extração dos caracteres de ambos os arquivos do formato binário (exe), traduzindo em formato hexadecimal através da ferramenta FlexHex [FlexHex].

Na Figura 3 é apresentado os cinco programas Jogo da Memória na ferramenta FlexHex e na Figura 4 é apresentado os dois programas de Calculadora na mesma ferramenta. É na ferramenta FlexHex que os códigos dos programas acima citados são transformamos do formato binário (exe) para hexadecimal [FlexHex].

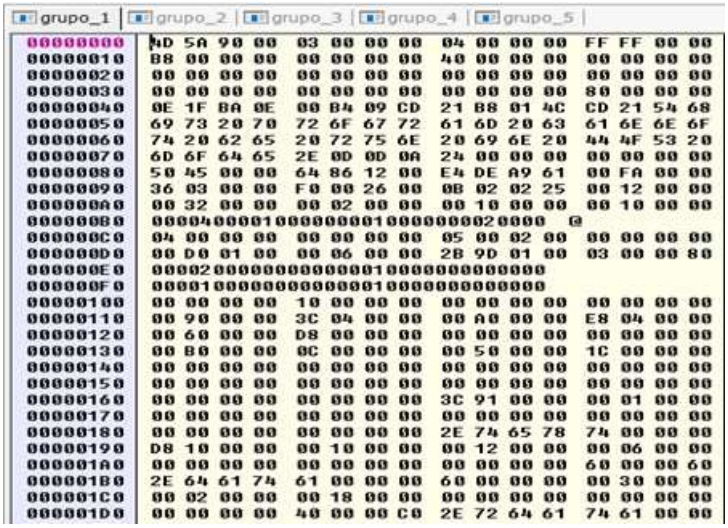


Figura 3. Exemplo dos cinco programas Jogo da Memória na ferramenta FlexHex

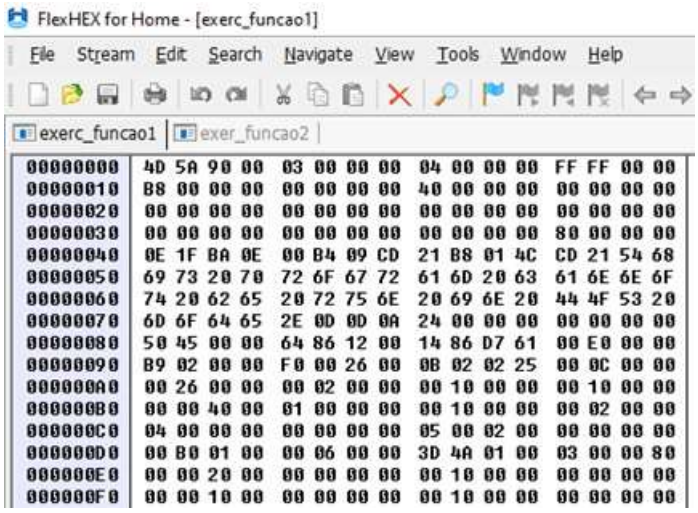


Figura 4. Exemplo dos dois programas de Calculadora na ferramenta FlexHex

4.2.1. Análise Estática – Comparação Textual (Verificação dos dados)

Os dados extraídos em formato hexadecimal dos cinco programas do Jogo da Memória e dos dois programas de Calculadora, no passo anterior, foram consolidados na ferramenta Microsoft Excel, sendo que o primeiro passo é colocar todos os resultados em hexadecimal dos programas numa mesma aba da planilha eletrônica, desta forma foram criadas duas abas, uma delas com todos os códigos em hexadecimal dos programas do Jogo da Memória e outra com os códigos dos programas de Calculadora. Após isso, todos os dados contidos na aba com os códigos em hexadecimal são selecionados e, usando-se a função Tabela Dinâmica da ferramenta Microsoft Excel, gera-se uma nova aba com uma tabela dos códigos em hexadecimal, na qual é possível ordenar os dados, fazer uma comparação entre as linhas dos códigos, a contagem das linhas de código e uma análise detalhada das similaridades e diferenças entre os programas.

Na Tabela 1 são apresentados todos os códigos hexadecimais e a quantidade de linhas repetidas por grupo dos cinco programas do Jogo da Memória e na Tabela 2 as mesmas informações são apresentadas para os dois programas da Calculadora.

Tabela 1. Códigos hexadecimais dos cinco programas Jogo da Memória na ferramenta Microsoft Excel.

Códigos Hexadecimais	Grupo1	Grupo2	Grupo3	Grupo4	Grupo5	Total Geral
01 04 01 00 04 42 00 00 01 00 00 00 01 00 00 0	1					1
01 10 04 85 10 03 08 01 20 00 01 50 01 08 03 0	1					1
08 72 04 03 01 50 00 00 01 04 01 00 04 42 00 0	1					1
01 00 00 00 01 00 00 00 01 00 00 00 01 00 00 0	1					1
01 06 03 00 06 42 02 30 01 60 00 00 01 00 00 0	1					1
01 04 01 00 04 62 00 00 01 00 00 00 01 00 00 0	1					1
01 00 00 00 01 08 03 00 08 42 04 C0 02 D0 00 0	1					1
48 83 EC 28 48 8D 0D 53 01 00 00 E8 B0 08 00 00		1				1
45 31 C0 31 D2 31 C9 E8 C4 08 00 00 45 31 C0 31		1				1
D2 31 C9 E8 C8 08 00 00 45 31 C0 31 D2 31 C9 E8		1				1
CC 08 00 00 45 31 C0 31 D2 31 C9 48 83 C4 28 E9		1				1
CC 08 00 00 90 90 90 90 90 90 90 90 90 90 90 90		1				1
4C 8D 05 A9 0F 00 00 31 D2 48 8D 0D 10 00 00 00		1				1
E9 4B 07 00 00 66 66 2E 0F 1F 84 00 00 00 00 00		1				1
C3 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90		6	6	6	6	24
55 48 89 E5 48 83 EC 70 C7 45 FC 00 00 00 00 EB		1				1

Tabela 2. Códigos hexadecimais dos dois programas de Calculadora na ferramenta Microsoft Excel.

Códigos Hexadecimais	ExerFuncao1	ExerFuncao2	Total Geral
48 83 EC 28 48 8D 0D 69 01 00 00 E8 10 04 00 00	1		1
45 31 C0 31 D2 31 C9 E8 24 04 00 00 45 31 C0 31	1		1
D2 31 C9 E8 28 04 00 00 45 31 C0 31 D2 31 C9 E8	1		1
2C 04 00 00 45 31 C0 31 D2 31 C9 48 83 C4 28 E9	1		1
2C 04 00 00 90 90 90 90 90 90 90 90 90 90 90 90	1		1
4C 8D 05 A9 0F 00 00 31 D2 48 8D 0D 10 00 00 00	1		1
E9 5B 03 00 00 66 66 2E 0F 1F 84 00 00 00 00 00	1		1
C3 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90	6	5	11
55 48 89 E5 48 83 EC 30 48 8D 05 71 1F 00 00 48	1		1
89 C1 E8 49 03 00 00 48 8D 45 FC 48 89 C2 48 8D	1		1
05 6F 1F 00 00 48 89 C1 E8 53 03 00 00 8B 45 FC	1		1

Após a geração da Tabela Dinâmica com os códigos em hexadecimais, uma nova aba é criada na ferramenta Microsoft Excel, na qual os dados da Tabela Dinâmica são copiados, gerando uma nova tabela em que, de fato, os resultados da comparação dos códigos entre si são obtidos, a fim de entendermos as suas similaridades.

Tabela 3. Comparação do percentual de similaridade entre os códigos hexadecimais dos cinco programas Jogo da Memória.

Comparações	G1 vs G2	G1 vs G3	G1 vs G4	G1 vs G5	G2 vs G3	G2 vs G4	G2 vs G5	G3 vs G4	G3 vs G5	G4 vs G5
Linhas 100% iguais	0	0	0	0	45	50	52	54	56	49
Qtd de Linhas diferentes	681	816	747	887	681	602	738	728	865	810
Total de linhas	681	816	747	887	726	652	790	782	921	859
Percentual de igualdade	0,0%	0,0%	0,0%	0,0%	6,2%	7,7%	6,6%	6,9%	6,1%	5,7%

Como é possível observar na Tabela 3, entre os cinco programas do Jogo da Memória, o Grupo 2 e o Grupo 4 são aqueles que possuem o maior percentual de igualdade textual, com

7,7% de similaridade entre os códigos hexadecimais. Nos demais Grupos, o percentual de similaridade entre os códigos é ainda menor, e por meio desta análise fica difícil comprovar qualquer indício claro de plágio.

Tabela 4. Comparação do percentual de similaridade entre os códigos hexadecimais dos dois programas de Calculadora.

Comparações	G1 vs G2
Linhas 100% iguais	133
Qtd de Linhas diferentes	134
Total de linhas	267
Percentual de igualdade	49,8%

Na Tabela 4, é apresentado o percentual de 49,8% de igualdade textual entre os códigos hexadecimais dos dois programas de Calculadora.

4.3. Engenharia Reversa (*Disassembler*)

A técnica de *disassembler* é fundamental para esse trabalho. Para usar essa técnica, é necessário fazer um processo de compilação, ou seja, conversão de um programa escrito em linguagem de alto nível para um programador em linguagem de baixo nível para ser executado pelo compilador.

> [...] Fazemos engenharia reversa quando queremos trocar, modificar uma peça (ou um software) por outro, com as mesmas características ou entender como esta funciona e não temos acesso a sua documentação [ENGENHARIA REVERSA 2005].

O motivo para uso dessa técnica é a necessidade de obtermos acesso ao código-fonte na forma com mais baixo nível, de modo que a comparação possa ser feita em diversos níveis do código. Portanto, como é possível observar na Figura 5, o *disassembler* é basicamente o ato de transformar o programa que está em binário (código de máquina) em uma linguagem de baixo nível.

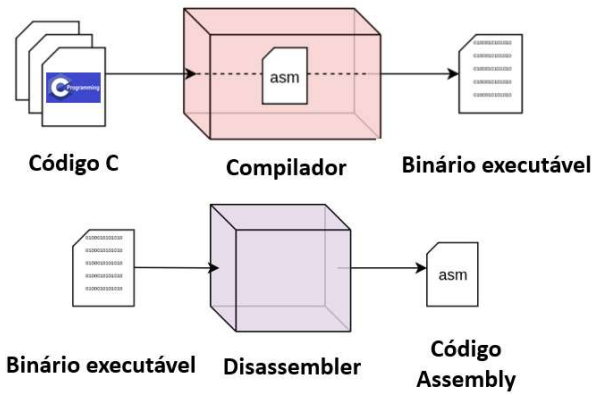


Figura 5. Exemplo do processo de *disassembler*

Desta forma, nesta busca por detalhes aprofundados do código internamente, suas funcionalidades e arquiteturas, temos, com o uso da engenharia reversa, uma maneira de prover a recuperação do código de máquina em linguagem *Assembly*, sendo possível analisar o funcionamento do programa através de quadros sequenciados de chamadas e suas linhas de código. Para essa análise, foi utilizada a ferramenta IDA Pro [IDA Pro] como é possível observar na Figura 6 essa ferramenta é um depurador iterativo, programável e escalável que rodamos no sistema operacional Windows.

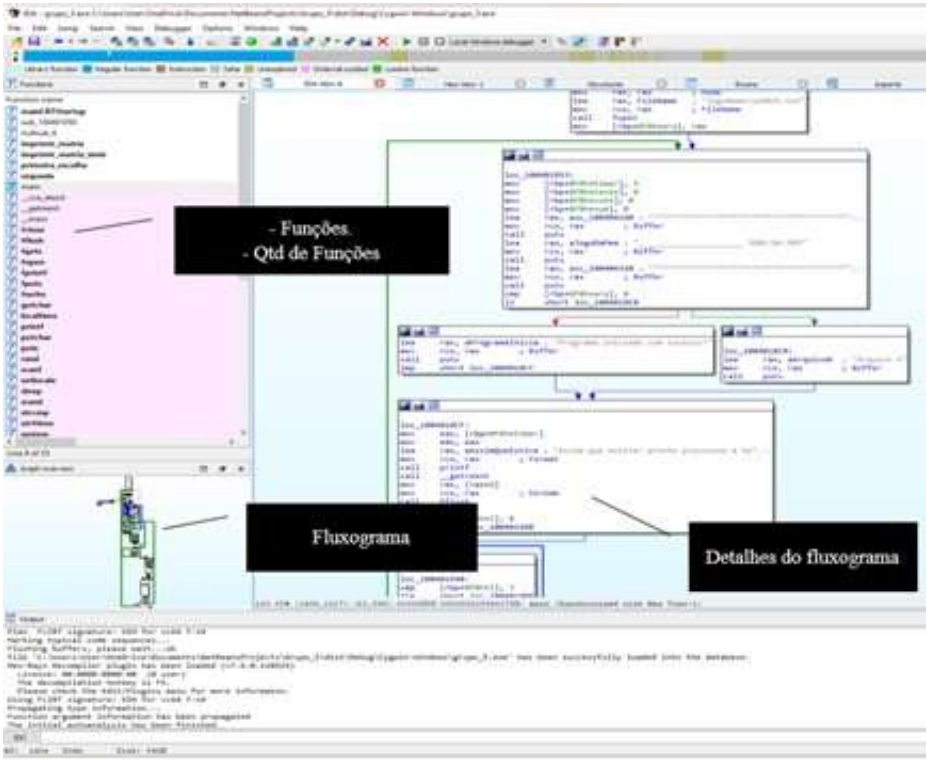


Figura 6. Exemplo de uso da ferramenta IDA Pro

4.3.1. Engenharia Reversa (*Disassembler*) - Funções

Através da ferramenta IDA Pro, é possível checar a quantidade de Funções por programa de cada um dos grupos do estudo de caso dos cinco programas do Jogo da Memória e dos dois programas de Calculadora [IDA Pro].

Na Figura 7 é possível observar a quantidade de Funções de cada grupo que fizeram os cinco programas Jogo da Memória e por meio desses dados foi feito a análise de quantidade de Funções de cada um dos cinco programas do Jogo da Memória, onde foram obtidos os seguintes resultados: o código-fonte do Grupo 1 possui 45 funções; no Grupo 2, ele possui 41 funções; no Grupo 3, possui 47 funções; no Grupo 4, possui 48 funções; no Grupo 5, possui 55 funções.

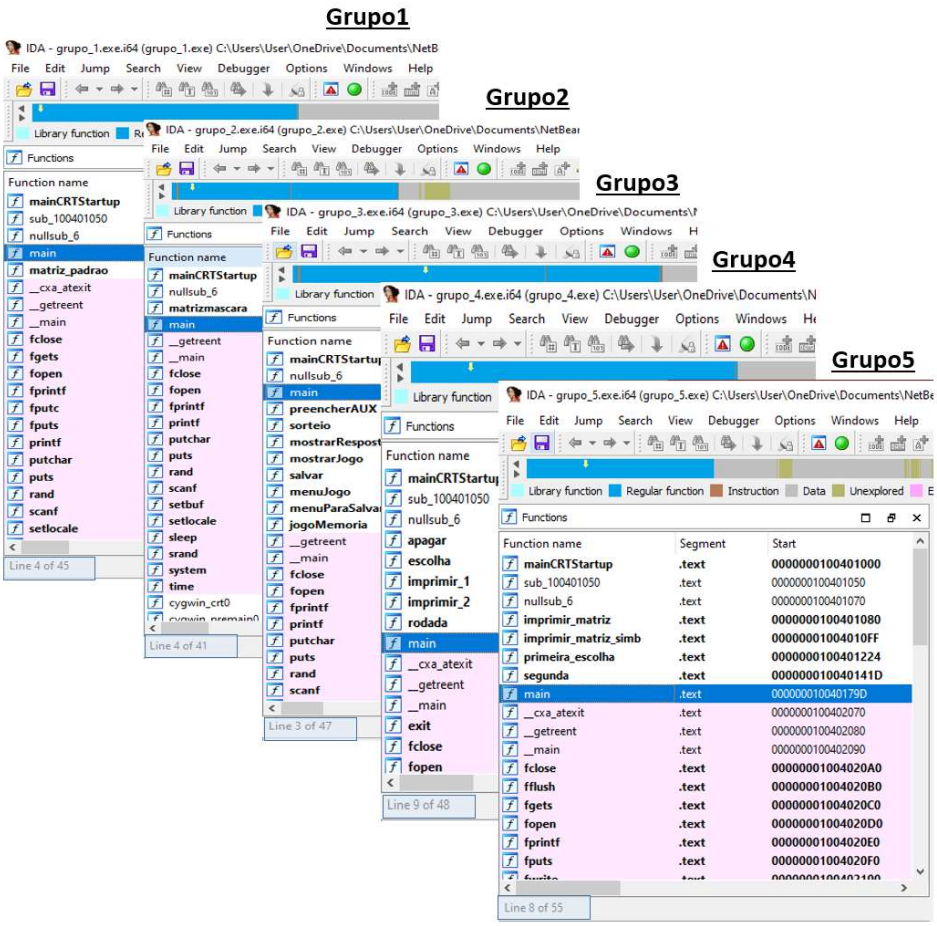


Figura 7. Detalhe das quantidades de Funções em cada grupo dos cinco programas Jogo da Memória

Muitas funções e chamadas são semelhantes entre os cinco programas Jogo da Memória, apesar das semelhanças, dado que os programas tinham o mesmo propósito, e tomando por base a função *main* de cada um dos programas, é possível observar que os programas começam em pontos diferentes na maioria dos casos, o que corrobora a caracterização de não plágio. A seguir, são listados os inícios do método *main* em cada grupo:

Grupo 1 – 0000000100401080

Grupo 2 – 000000010040115E

Grupo 3 – 0000000100401080

Grupo 4 – 000000010040139E

Grupo 5 – 000000010040179D

Apesar de o Grupo 1 e o Grupo 3 começarem o método *main* na mesma função (0000000100401080), uma vez observada toda estrutura do código na ferramenta IDA Pro [IDA Pro], é possível verificar que o Grupo 3 possui funções totalmente diferentes do Grupo 1,

logo após o *main*, por exemplo, as funções preencherAUX, sorteio, mostrarReposta, mostrarJogo, entre outras, conforme Figura 7.

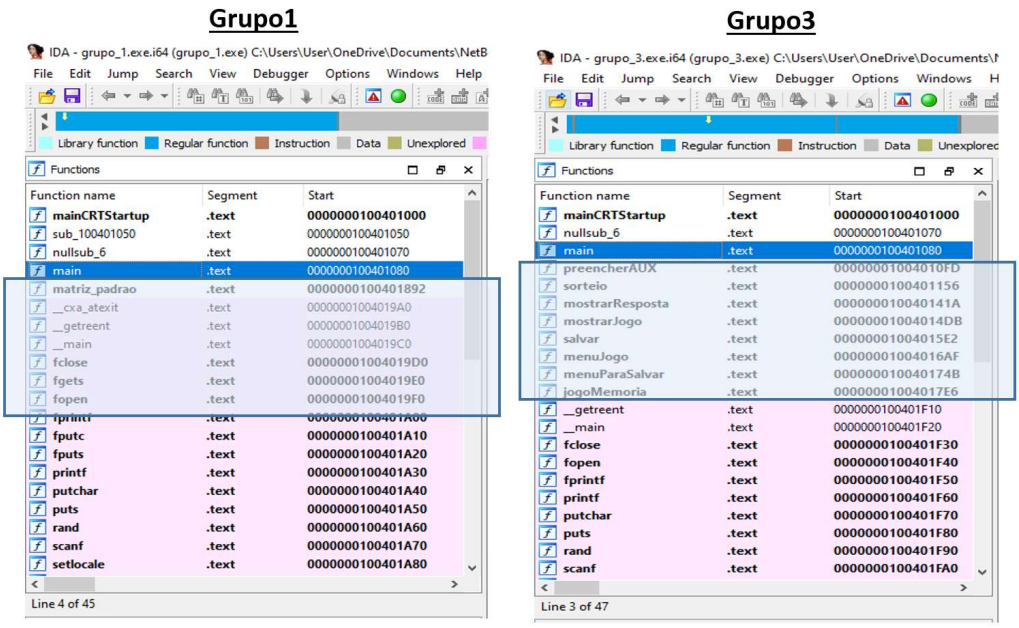


Figura 8. Detalhe da função *Main* entre o Grupo 1 e o Grupo 3

A checagem da quantidade de Funções por programa, para o estudo de caso dos dois programas de Calculadora não foi necessária, uma vez que, por se tratarem de dois programas idênticos e com apenas as *Strings* diferentes, ou seja, um deles todo escrito no idioma português e o outro todo escrito no idioma inglês, as funções são as mesmas, o que resultaria em uma igualdade entre os programas.

4.3.2: Análise Estática – Comparação Textual (*Strings*)

A ferramenta IDA Pro também permite que sejam visualizadas as Strings de cada programa, como pode-se observar na Figura 9 onde são apresentadas as Strings dos cinco programas de Jogo da Memória e na Figura 10 onde são apresentadas as Strings dos dois programas de Calculadora [IDA Pro].

O intuito dessa análise é identificar uma sequência de caracteres (textos) entre os programas e, desta forma, poder inferir sobre a existência de plágio, por isso essa análise é realizada para os cinco programas do Jogo da Memória e os dois programas de Calculadora, para isso é realizada a extração dos resultados das Strings de cada um dos programas na ferramenta IDA Pro e consolidados na ferramenta Microsoft Excel [IDA Pro].

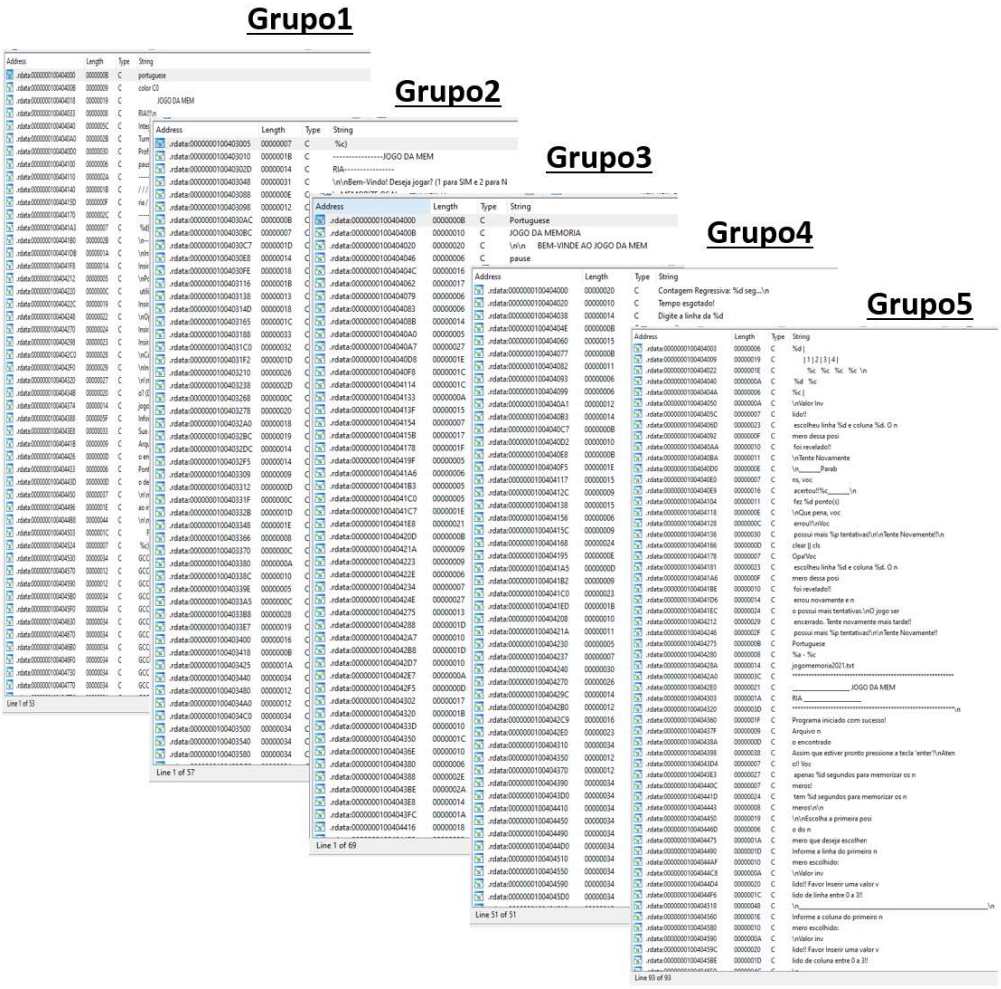


Figura 9. ***Strings*** **obtidas pela ferramenta IDA Pro dos cinco programas de Jogo da Memória**

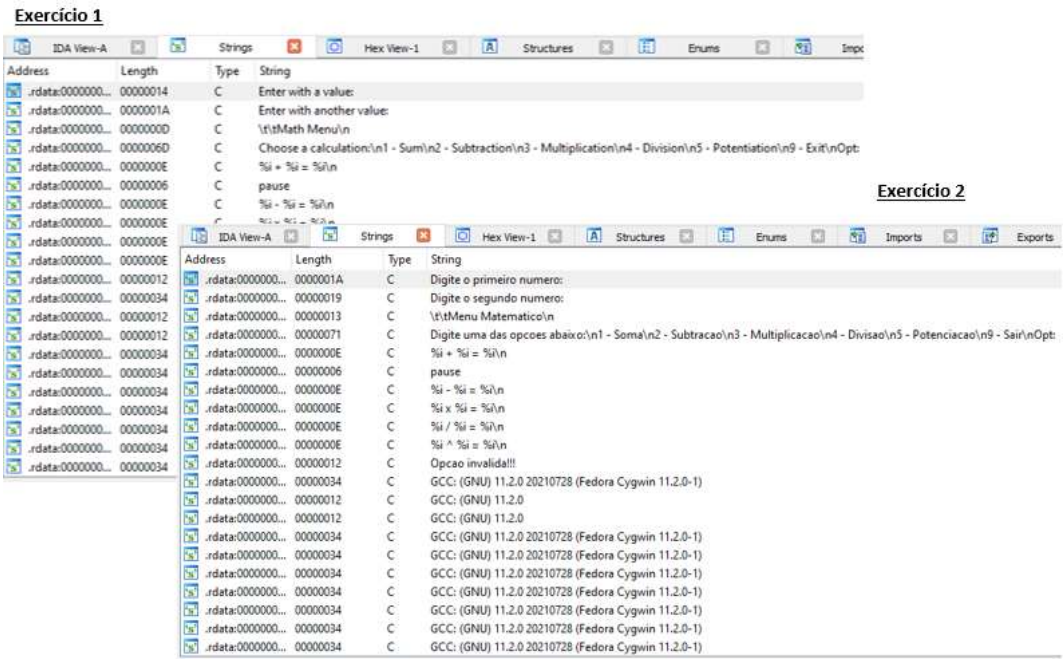


Figura 10. ***Strings*** **obtidas pela ferramenta IDA Pro dos dois programas de Calculadora**

O processo para geração e comparação dos resultados das *Strings*, na ferramenta Microsoft Excel, é exatamente o mesmo feito no passo Análise Estática – Comparação Textual (Verificação dos dados), em que o primeiro passo na ferramenta Microsoft Excel é colocar todos os resultados dos programas numa mesma aba da planilha eletrônica. Desta forma, são criadas duas abas, uma delas com todos as *Strings* dos programas do Jogo da Memória e outra com os dois programas de Calculadora, após isso, todos os dados contidos em cada uma das abas com as *Strings* são selecionados, e usando-se a função Tabela Dinâmica, da ferramenta Microsoft Excel, é gerada uma nova aba com uma tabela com as *Strings*, na qual é possível ordenar os dados, fazer uma comparação entre as linhas das *Strings*, a contagem das linhas e uma análise detalhada das similaridades e diferenças entre os programas.

Após realizar a análise comparativa entre os códigos-fontes dos cinco programas do Jogo da Memória, o resultado obtido demonstra um percentual baixo de repetições de *String*. Várias repetições são *Strings* geradas pelo compilador GNU, logo, fica caracterizado que não houve plágios entre as *Strings* dos cinco programas.

Tabela 5. Compilação de todas as *Strings* dos cinco programas do Jogo da Memória

Strings dos programas	Grupo1	Grupo2	Grupo3	Grupo4	Grupo5	Total Geral
FINALIZANDO O JOGO...	1					1
%c |			1			1
%c)	1	1				2
%d %c					1	1
%d |			1			1
%d)	1	1				2
a pr			1			1
ACABANDO!		1				1
Acertou =)\nPr				1		1
acertou!!%c_____\n					1	1
apenas %d segundos para memorizar os n					1	1
encerrado. Tente novamente mais tarde!!					1	1
errou =(\nTente novamente:				1		1
errou 10 vezes.				1		1
errou novamente e n					1	1
errou!!\nVoc					1	1
escolha:				2		2
escolheu linha %d e coluna %d. O n					2	2
fez %d ponto(s)					1	1
foi revelado!!					2	2

Tabela 6. Percentual de similaridade das *Strings* dos cinco programas do Jogo da Memória

Comparações	G1 vs G2	G1 vs G3	G1 vs G4	G1 vs G5	G2 vs G3	G2 vs G4	G2 vs G5	G3 vs G4	G3 vs G5	G4 vs G5
Linhas 100% iguais	5	5	5	6	3	3	4	5	6	4
Qtd de Linhas diferentes	72	83	67	97	89	73	102	80	110	98
Total de linhas	77	88	72	103	92	76	106	85	116	102
Percentual de igualdade	6,5%	5,7%	6,9%	5,8%	3,3%	3,9%	3,8%	5,9%	5,2%	3,9%

Para os dois programas de Calculadora, após realizar a análise comparativa entres os códigos-fontes, mais uma vez os resultados obtidos demonstram um percentual baixo de repetições de *Strings*. Entre as várias repetições, observa-se que se tratam de *Strings* geradas pelo compilador GNU, mais uma vez evidenciando que apenas essa análise é insuficiente para caracterizar indícios de plágios entre as *Strings* dos dois programas de Calculadora.

Tabela 7. Compilação de todas as *Strings* dos dois programas de Calculadora

Grupos	Strings
ExerFuncao1	Enter with another value:
ExerFuncao1	Enter with a value:
ExerFuncao1	Choose one of the alternatives:\n1 - Addition\n2 - Subtraction\n3 - Multiplication\n4 - Division\n5 - Potentiation\n6 - Exit\nOpt:
ExerFuncao1	%i x %i = %i\n
ExerFuncao1	%i ^ %i = %i\n
ExerFuncao1	%i / %i = %i\n
ExerFuncao1	%i - %i = %i\n
ExerFuncao1	%i + %i = %i\n
ExerFuncao2	Digite o primeiro numero:
ExerFuncao2	Digite o segundo numero:
ExerFuncao2	\t\tMenu Matematico\n
ExerFuncao2	Digite uma das opcoes abaixo:\n1 - Soma\n2 - Subtracao\n3 - Multiplicacao\n4 - Divisao\n5 - Potenciacao\n9 - Sair\nOpt:
ExerFuncao2	%i + %i = %i\n
ExerFuncao2	pause
ExerFuncao2	%i - %i = %i\n
ExerFuncao2	%i x %i = %i\n
ExerFuncao2	%i / %i = %i\n
ExerFuncao2	%i ^ %i = %i\n

Tabela 8. Percentual de similaridade das *Strings* dos dois programas de Calculadora

Comparações	G1 vs G2
Linhas 100% iguais	8
Qtd de Linhas diferentes	10
Total de linhas	18
Percentual de igualdade	44,4%

4.3.3. Comparação Textual – Via ferramenta *online Copyleaks*

Como forma de averiguar os resultados dessa análise do uso combinado de técnicas de análise estática, foi feita uma comparação dos resultados obtidos até então com os resultados de uma ferramenta antiplágio *online* chamada *Copyleaks* [Copyleaks].

O processo aqui é simples, como tínhamos acesso ao código-fonte de cada programa, foi feita a comparação de cada um dos programas entre si por meio da ferramenta *Copyleaks*. Os resultados detalhados podem ser consultados no Anexo A [Copyleaks].

Quanto aos códigos-fontes dos cinco programas do Jogo da Memória, os resultados obtidos foram compilados por meio da ferramenta Microsoft Excel, conforme pode ser observado na Figura 9, mais uma vez fica evidente, por si só, esses dados não indicam haver um plágio entre os grupos, uma vez que apenas o Grupo 1 e o Grupo 2 apresentaram algum tipo de similaridade, ainda assim com um percentual muito pequeno, de 11,7%.

Tabela 9. Percentual de similaridade das Strings, via ferramenta Copyleaks, dos cinco programas do Jogo da Memória

Comparações	G1 vs G2	G1 vs G3	G1 vs G4	G1 vs G5	G2 vs G3	G2 vs G4	G2 vs G5	G3 vs G4	G3 vs G5	G4 vs G5
Idêntico	2,8%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
Pequenas Alterações	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
Significado Relacionado	9,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
Palavras Omitidas	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
Percentual de igualdade	**11,7%**	**0,0%**	**0,0%**	**0,0%**	**0,0%**	**0,0%**	**0,0%**	**0,0%**	**0,0%**	**0,0%**

Para os códigos-fontes dos dois programas de Calculadora, os resultados obtidos via Copyleaks demonstram uma combinação de 100% entre os dois programas, sendo importante ressaltar que, apesar de um programa estar 100% no idioma português e outro no idioma inglês, a ferramenta *Copyleaks* possui uma grande vantagem com relação a muitos concorrentes, pois a mesma faz uso de Inteligência Artificial, por isso é capaz de reconhecer como "Pequenas Alterações" as mudanças de idiomas entre textos [Copyleaks].

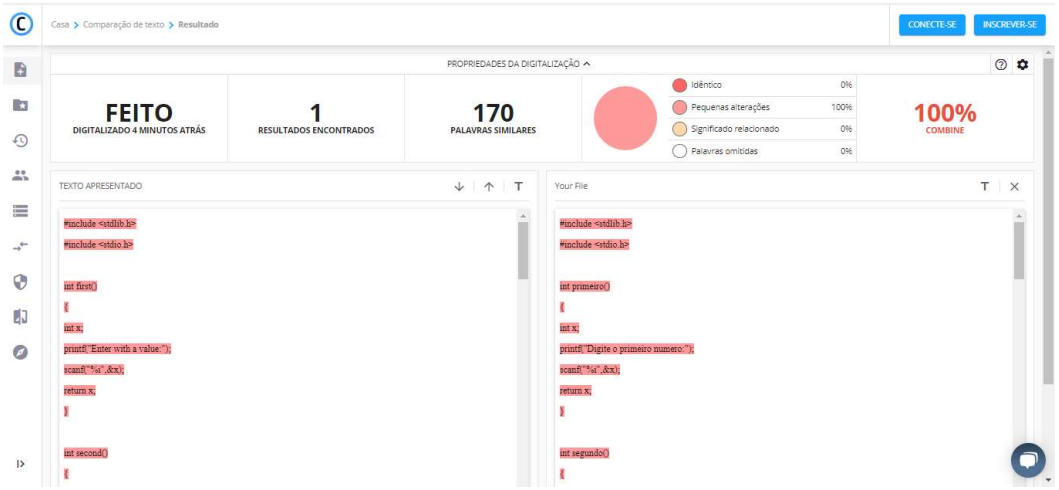


Figura 11. Percentual de similaridade das *Strings*, via ferramenta *Copyleaks*, dos dois programas de Calculadora

Porém, um problema da *Copyleaks* é que, apenas mudando as ordens do texto, ela passa a não reconhecer mais as similaridades, ou seja, a ferramenta apresenta bons resultados apenas se os textos comparados estiverem na mesma posição um do outro, pois basta que uma linha de código seja mudada de posição, e a ferramenta não detecta mais a similaridade [Copyleaks].

Veja abaixo um exemplo dessa falha, onde é simulada uma troca de ordem do bloco inicial escrito do programa Calculadora no idioma inglês, e o outro programa no idioma português permanece original, ou seja, em um dos programas as primeiras linhas de código são movidas para outra posição do código.

Notem na figura abaixo que todo o bloco mudado passa a ficar irreconhecível, e o percentual, que antes era de 100% de combinação com "Pequenas Alterações", passa a ser de 74,1%. Dentro dos 74,1% combinação, temos a seguinte divisão 72,9% indica que está totalmente idêntico um código do outro; 0,6% possui pequenas alterações e 0,6% de significados relacionados.

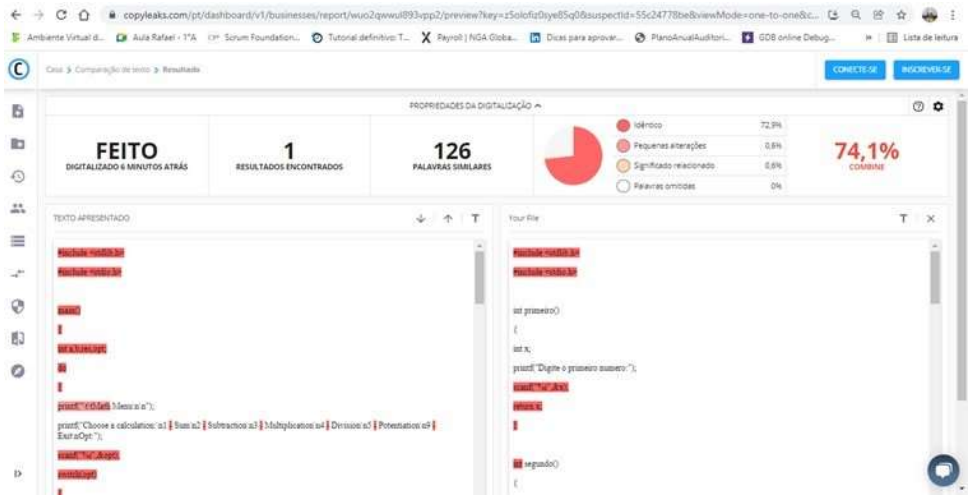


Figura 12. Percentual de similaridade das *Strings* via ferramenta *Copyleaks* dos dois programas de Calculadora com mudanças na ordem do código

Conseguimos com essa simulação demonstrar que essas ferramentas que fazem comparação de texto são adequadas para literatura, mas não adequadas para análise de plágio de SW em códigos-fontes, uma vez que elas não levam em conta a posição do código, assumindo nesse caso que os programas são diferentes. Portanto, são limitadas, pois uma simples mudança no nome de um método faz a ferramenta perder a análise e não evidenciar um plágio onde realmente pode haver.

4.3.4. Engenharia Reversa (*Disassembler*) - Gráfico das Arquiteturas

A ferramenta IDA Pro possui uma função gráfica que apresenta toda a visão geral da arquitetura dos programas, logo é possível ver todas as interfaces de cada chamamento entre as funções e entender como cada sessão do código interage/acessa a outra por meio de um gráfico, ou seja, ela gera um fluxograma do código inteiro que está sendo analisado, e cada cor tem uma função dentro da arquitetura do programa [IDA Pro].

Com a análise do fluxo de cada um dos programas, é possível entender a representação da codificação e evidenciar um plágio. Após executar o programa IDA Pro, é possível obter as arquiteturas de cada um dos programas [IDA Pro].

Na Figura 13 pode-se verificar a representação gráfica de cada um dos fluxos dos cinco programas do Jogo da Memória e observe que, entre o Grupo 3 e Grupo 4, os gráficos possuem um código com menos fluxos, ou seja, isso representa um código mais alinhado, ao passo que na relação entre o Grupo 1 e o Grupo 2 existe uma programação também similar, porém com muitas chamadas e métodos, e por fim o Grupo 5 aparece como um intermediário entre esses grupos.

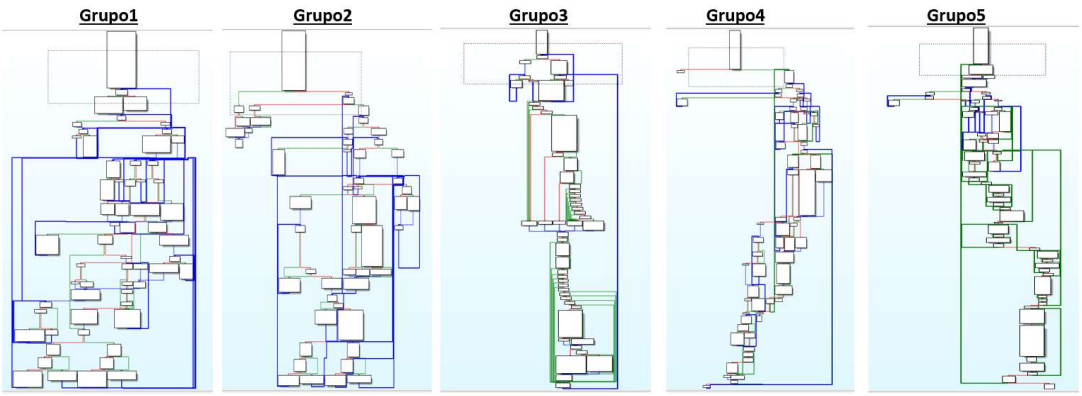


Figura 13. Comparação gráfica das arquiteturas de cada um dos cinco programas

Ainda nesta análise de similaridade das arquiteturas dos cinco programas do Jogo da Memória, é possível fazer uma análise dos gráficos baseada nos códigos das funções, bem como o fluxo de chamamento da árvore, para isso, partimos da principal função, chamada de função *Main*, de cada um dos cinco grupos. Escolhemos essa função já que ela é comum entre todos os programas.

Observa-se na Figura 14, que os pontos de chamamentos, a partir do *Main* de cada um dos cinco grupos, são bem diferentes, o que demonstra que cada um dos projetos possui seu próprio estilo de programação.

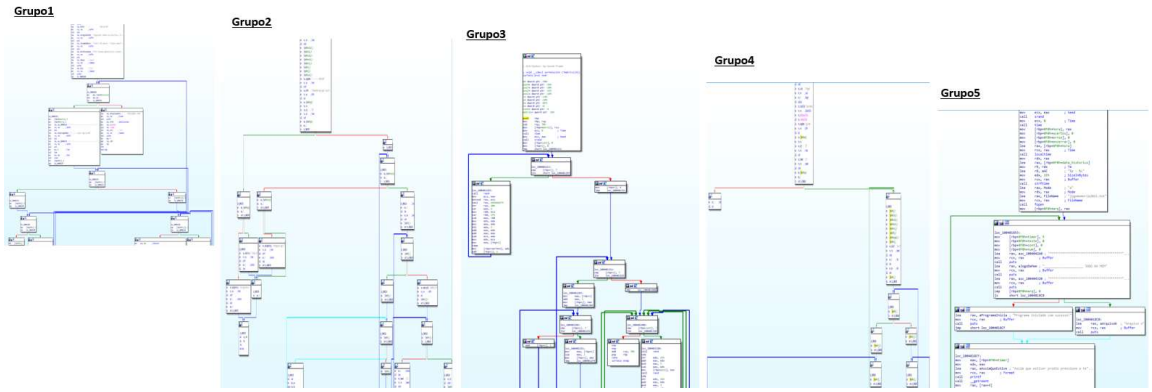


Figura 14. Apresentação gráfica das arquiteturas dos cinco programas de Jogo da Memória a partir da função *Main*

Com relação aos dois programas de Calculadora, conforme pode ser observado na Figura 15, onde é possível fazer uma análise gráfica da visão geral da arquitetura dos programas, evidenciasse que todas as interfaces de cada chamamento entre as funções e o modo como cada sessão do código interage entre si são idênticos, o que caracterizaria a existência de plágio. Vale relembrar que esses dois programas que simulam uma Calculadora são similares, sendo que a diferença entre eles é que um está todo escrito em português e o outro todo escrito em inglês.

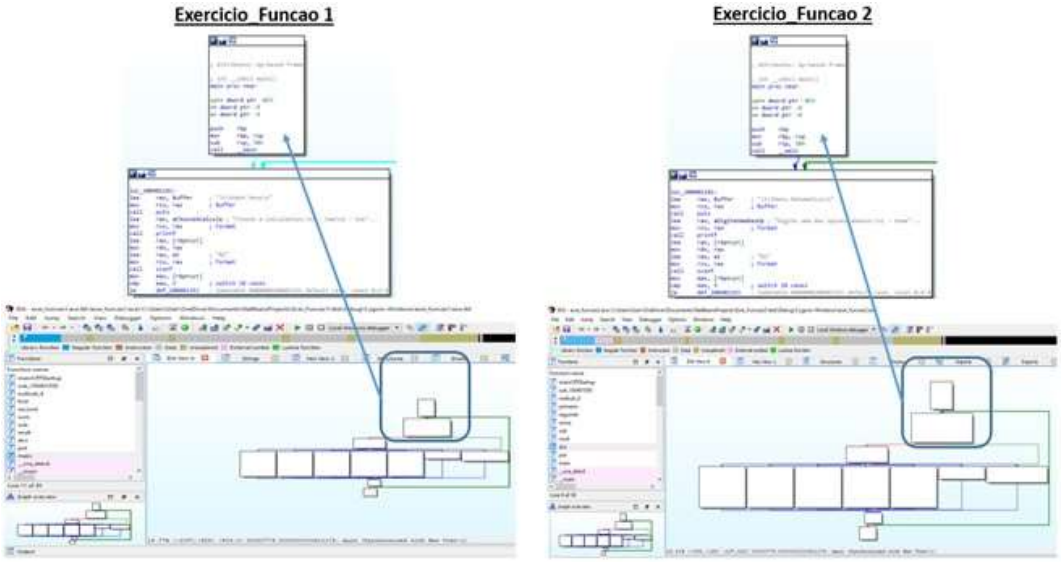


Figura 15. Apresentação gráfica das arquiteturas dos dois programas de Calculadora

Desta forma, fica evidente a efetividade da análise da visão geral das arquiteturas dos programas, para inferir sobre um possível plágio de *software*, pois o gráfico demonstrou com clareza como os códigos eram idênticos em quantidade e esquema de funções e chamamentos de suas rotinas para o caso dos dois programas de Calculadora.

5. Conclusões

É inegável que o plágio de *software* é um problema crônico no mundo em que vivemos, afetando desde grandes corporações públicas e privadas até o meio acadêmico. Por isso, a Lei do *Software* nº 9.609/98 e seu regulamento, Decreto nº 2.556/98, com a "Lei de Direito Autoral" nº 9.610/98, vieram para ajudar a proteger a propriedade intelectual, entretanto, é importante frisar que apenas indícios de que ocorreu plágio não são suficientes para serem usados como prova, em caso de alguma denúncia. Provar que ocorreu um plágio não é uma tarefa trivial, pois necessita de uma série de técnicas estruturadas, o que por sua vez demanda mão de obra especializada.

Neste trabalho, foi aplicada a combinação das técnicas de análise estática (similaridade sintática e semântica) nos cinco grupos que fizeram o programa chamado Jogo da Memória, na disciplina de Linguagem de Programação I, no Instituto Federal de Educação, Ciência e Tecnologia de São Paulo, Campus de Hortolândia. Nesta perícia, os grupos apresentaram poucos indícios de haver um plágio de *software* entre eles, pois na combinação das técnicas de análise estática, ao longo do estudo de caso, comprovou-se que a correlação entre as funções dos programas, do fluxograma do código e seus chamamentos, a análise das *Strings* e do código-fonte em hexadecimal, é muito distinta entre os grupos, o que evidencia a ausência de plágio entre os programas.

Por fim, mas não menos importante, na análise feita em paralelo para os dois programas que simulam uma Calculadora, em que um deles está todo escrito em português e o outro todo escrito em inglês, corroborou-se a afirmação de que o uso combinado de técnicas de análise

estática para identificação de plágio é mais vantajoso que o uso isolado. Nesse caso, a aplicação combinada das técnicas de análise estática mostrou-se mais recomendável para análise de plágio de *software*, uma vez que o uso isolado de uma ferramenta para identificação de plágio, no caso, uma ferramenta que faz uso da similaridade sintática, apresentou bons resultados apenas quando o plágio foi algo descarado, totalmente igual na ordem dos códigos e na escrita dos mesmos, ao passo que, se o plagiador fizer uma simples mudança na ordem do código, fatalmente o programa apresentará resultados que demonstram não haver plágio.

Logo, a combinação dessas técnicas de similaridade sintática e semântica é capaz de identificar indícios de um possível plágio mais elaborado, pois a análise mais robusta no código, combinando diferentes técnicas e ferramentas, permite essa averiguação.

Por fim, como trabalhos futuros sugere-se a implementação das técnicas de similaridade sintática e semântica em programas mais robustos, ou seja, códigos que fazem uso de linguagens orientadas a objetos, com banco de dados e interface gráfica mais madura, ou seja, em códigos desenvolvidos nas disciplinas mais avançadas do curso de Análise e Desenvolvimento de *Software* do Instituto Federal de São Paulo, no campus Hortolândia.

Para o desenvolvimento deste trabalho, e até mesmo destacando as competências e habilidades adquiridas no decorrer do curso superior de Tecnologia em Análise e Desenvolvimento de Sistemas, houve aplicação de conceitos associados às disciplinas de Engenharia de *Software*, Linguagem de Programação, Metodologia de Pesquisa e Projeto de Sistemas para planejamento e desenvolvimento teórico-técnico.

6. Referências

Monteiro, M., Gorayeb I. L.; Monteiro, R., Neto, E. C., Noleto, E., (2018). "Análise Comparativa para Identificação de Plágio de Software". Editora Leud, 1ª edição.

We Are Social e HootSuite - Digital 2021 [Resumo e Relatório Completo]. https://www.amper.ag/post/we-are-social-e-hootsuite-digital-2021-resumo-e-relat%C3%B3rio-completo. [Online: acessado em 25 de novembro de 2021].

Silva, A. K. L. D., Domingues, M. J. C. D. S. (2008) Plágio no Meio Acadêmico: percepção dos alunos de pós-graduação sobre o tema. VI Simpósio de Gestão e Estratégia em Negócios.

Culwin, F. and Lancaster, T. (2000). A review of electronic services for plagiarism detection in student submissions. In LTSN-ICS 1st Annual Conference.

Lei nº 9.609. http://www.planalto.gov.br/ccivil_03/leis/l9609.htm. [Online: acessado em 12 de agosto de 2021].

Dissecting Go Binaries. Disponível em: https://www.grant.pizza/blog/dissecting-go-binaries/. [Online: acessado em 10 de novembro de 2021].

Uso do Algoritmo Distância de Edição com Técnicas de Pré-Processamento para Apoiar a Identificação de Plágio em Códigos-Fonte de Problemas de Programação Introdutória, https://sol.sbc.org.br/journals/index.php/isys/article/view/308. [Online: acessado em 10 de agosto de 2021].

Mota. M. A.; Goya. H. D. “Técnicas para Análise de Similaridade de Código de Software e Litígios de Propriedade Intelectual”. https://www.ime.usp.br/~dhgoya/forense_paper.pdf. [Online: acessado em 21 de agosto de 2021].

Junior A. J. S. C.; Souza D. A,; Moutinho D. S.; Lohnefink F. P (2005). “Engenharia Reversa”. http://www2.ic.uff.br/~otton/graduacao/informaticaI/apresentacoes/eng_reversa.pdf. [Online: acessado em 25 de novembro de 2021].

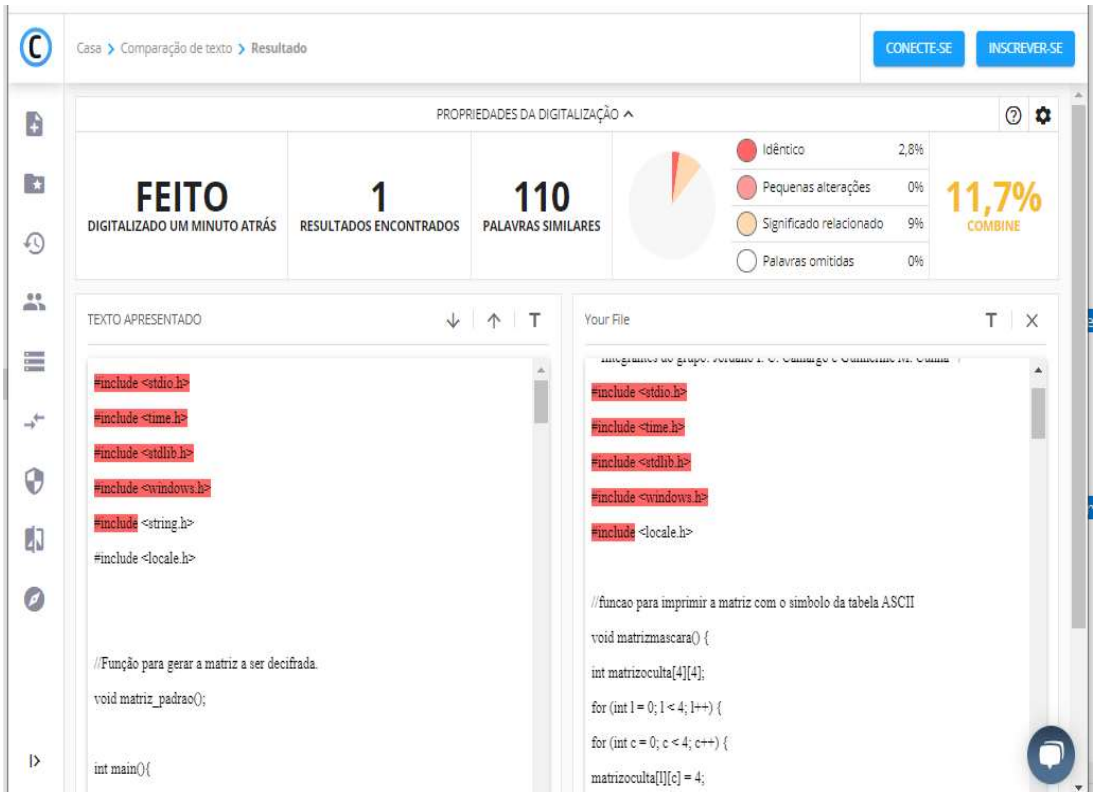
IDA Pro - Hex Rays. https://hex-rays.com/ida-pro/. [Online: acessado em 20 de agosto de 2021].

FlexHex - https://www.flexhex.com/. [Online: acessado em 15 de agosto de 2021].

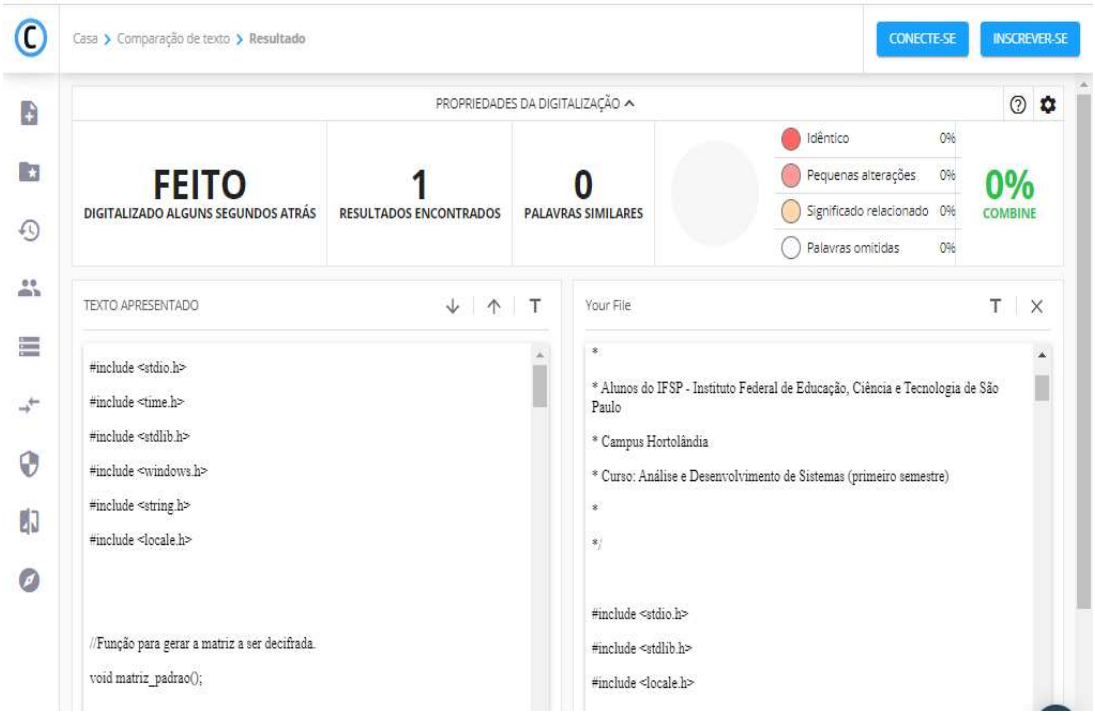
Copyleaks - https://copyleaks.com/pt/. [Online: acessado em 25 de agosto de 2021].

7. ANEXO A - COMPARAÇÃO TEXTUAL VIA FERRAMENTA *COPYLEAKS*

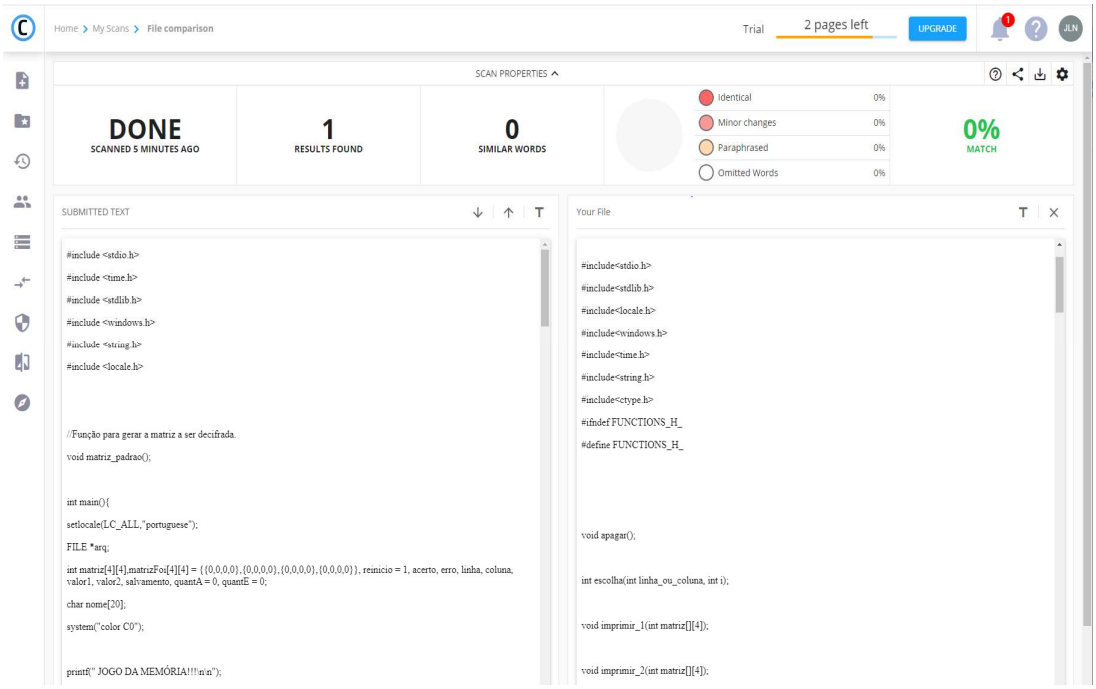
Grupo 1 comparado com Grupo 2



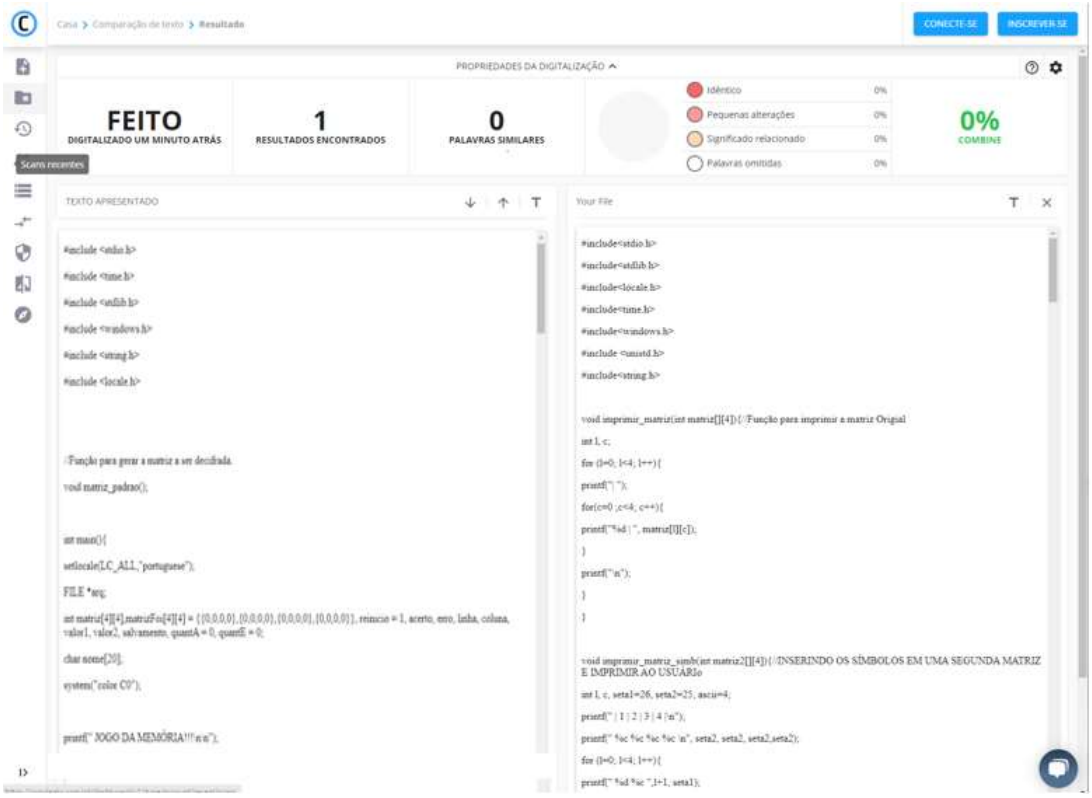
Grupo 1 comparado com Grupo 3



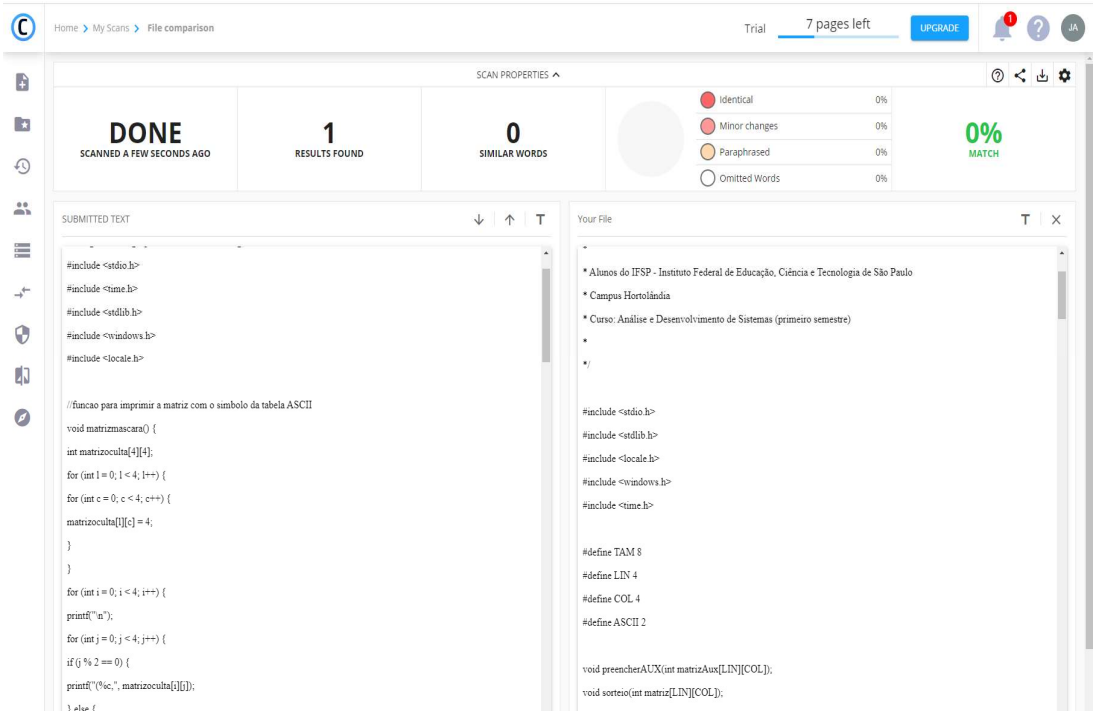
Grupo 1 comparado com Grupo 4



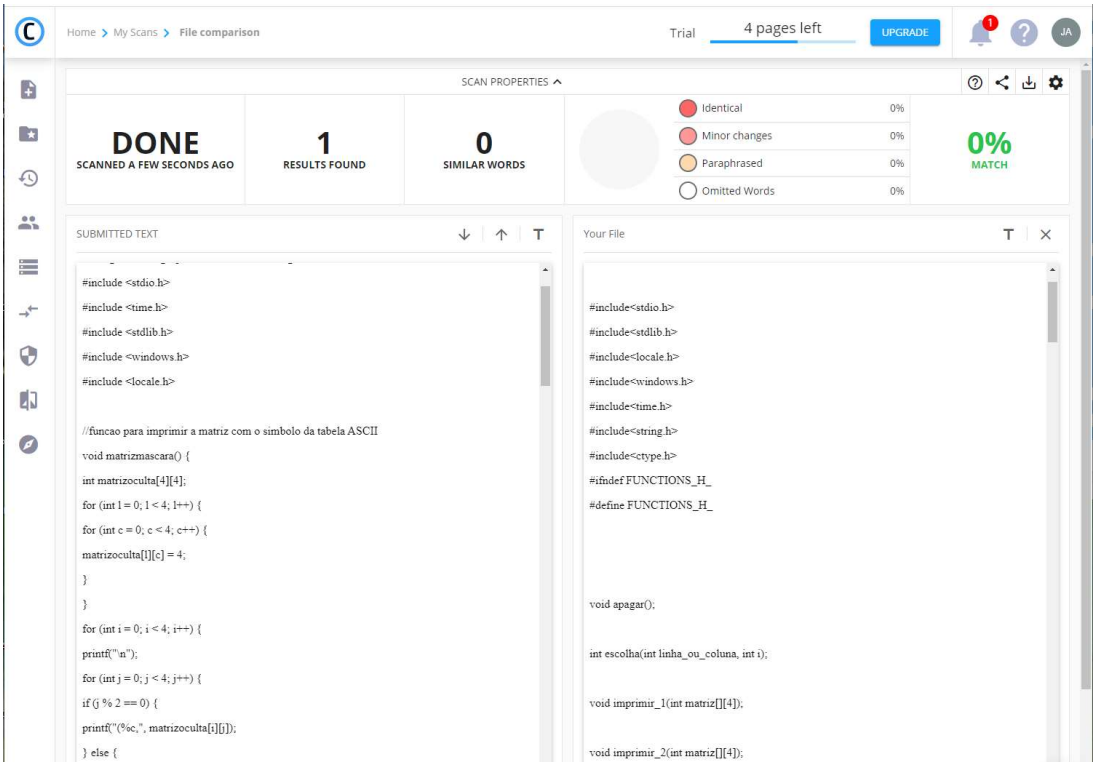
Grupo 1 comparado com Grupo 5



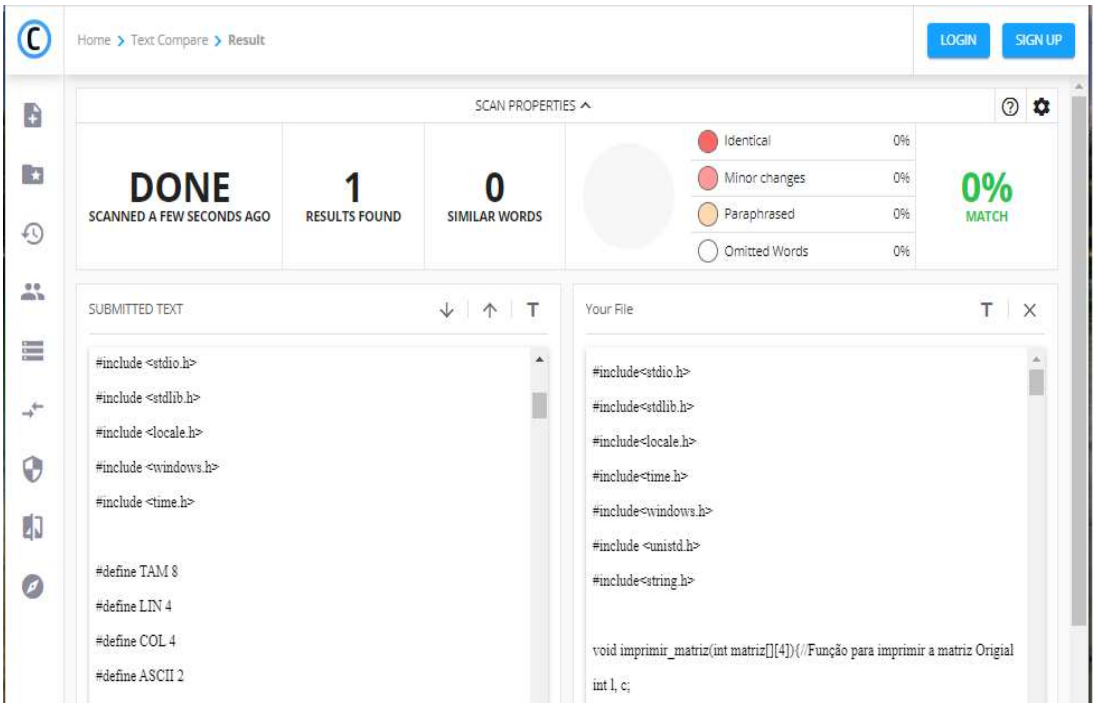
Grupo 2 comparado com Grupo 3



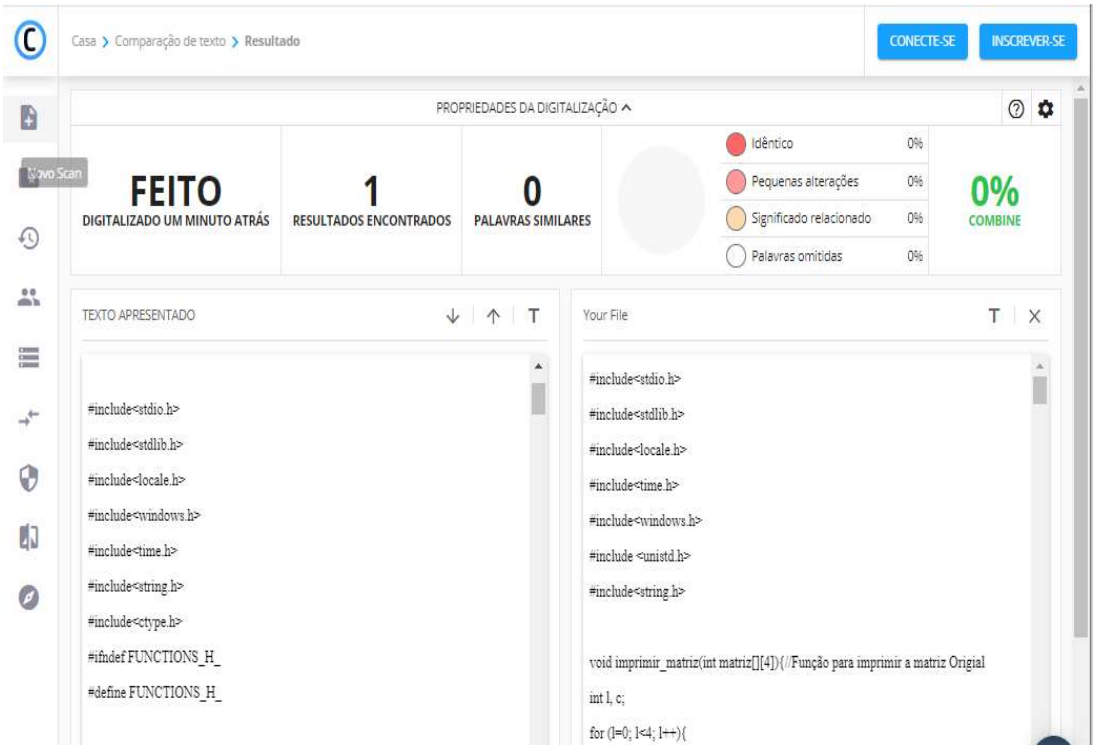
Grupo 2 comparado com Grupo 4



Grupo 3 comparado com Grupo 5



Grupo 4 comparado com Grupo 5



Documento Digitalizado Público

Versão Final do Artigo de TCC

Assunto: Versão Final do Artigo de TCC
Assinado por: Fernando Sambinelli
Tipo do Documento: Anexo
Situação: Finalizado
Nível de Acesso: Público
Tipo do Conferência: Documento Digital

Documento assinado eletronicamente por:

- **Fernando Sambinelli**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 10/06/2022 15:30:04.

Este documento foi armazenado no SUAP em 10/06/2022. Para comprovar sua integridade, faça a leitura do QRCode ao lado ou acesse https://suap.ifsp.edu.br/verificar-documento-externo/ e forneça os dados abaixo:

Código Verificador: 1005205
Código de Autenticação: f09860eccc