PortuCol: uma pseudolinguagem inspirada em C ANSI para o Ensino de Lógica de Programação e Algoritmos

Lucas Lemos Barbosa, Ricardo Terra

Departamento de Ciência da Computação,
Universidade Federal de Lavras (UFLA), Brasil

lbarbosa@computacao.ufla.br,terra@dcc.ufla.br

Resumo. *Linguagens de programação são intrinsecamente complexas, devido à variabilidade, generalidade e completude de suas construções. Isso justifica a adoção de pseudolinguagens de alto nível no ensino de Lógica de Programação e Algoritmos. Embora C ANSI e Java estejam entre as linguagens de programação mais utilizadas, Portugol – uma pseudolinguagem com instruções em português inspirada na linguagem obsoleta Pascal – é ainda largamente adotada por Instituições de Ensino Superior (IES). Diante desse cenário, este artigo propõe PortuCol, uma pseudolinguagem de programação com instruções em português, assim como Portugol, porém inspirada em C ANSI. Este artigo também descreve PortuCol2C, uma ferramenta que traduz código escrito em PortuCol para C ANSI e conduz estudos empíricos, que demonstram, estatisticamente, que (i) PortuCol é mais similar a C ANSI do que o Portugol e (ii) PortuCol é para C ANSI o que Portugol é para Pascal.*

Abstract. *Programming languages are intrinsically complex due to the variability, generality, and completeness of their constructions. This justifies the adoption of high-level pseudocode languages when teaching Introduction to Algorithms and Programming Logic. Although ANSI C and Java are among the most used programming languages, Portugol—a pseudocode language with instructions in Portuguese, inspired by the outdated Pascal language—is still widely adopted by Brazilian universities. In view of such circumstances, this paper proposes PortuCol, a pseudocode programming language with instructions in Portuguese, as Portugol, although inspired by ANSI C. It also describes PortuCol2C, a tool that translates PortuCol to ANSI C, and conducts empirical studies, which statistically demonstrate that (i) PortuCol is more similar to ANSI C than Portugol and (ii) PortuCol is for C ANSI what Portugol is for Pascal.*

1. Introdução

No início do ensino superior, é comum que estudantes da área de Ciência da Computação e afins tenham dificuldade no raciocínio lógico para criação de soluções, uma vez que, no ensino médio, esses estudantes estavam acostumados a apenas aplicar fórmulas previamente existentes para a solução de problemas. Um estudo realizado em [3] atesta essa dificuldade, reportando que, em média, mais de 50% dos alunos das disciplinas de introdução à programação são reprovados a cada semestre.

Em virtude de linguagens de programação serem intrinsecamente complexas – devido à variabilidade, generalidade e completude de suas construções – docentes de Instituições de Ensino Superior (IES), usualmente, adotam pseudolinguagens de alto nível nas disciplinas de introdução à Lógica de Programação e Algoritmos. Pseudolinguagens proporcionam ao aluno um maior espaço para o desenvolvimento de seu raciocínio lógico relacionado à resolução de problemas, sem preocupações relativas ao uso correto de uma linguagem de programação.

Portugol é, hoje no Brasil, a pseudolinguagem mais adotada por universidades com esse propósito [12, 14], a qual conta com diversas ferramentas de apoio ao ensino [5, 7, 12, 10, 6]. No entanto, Portugol é o produto de uma simbiose do português e as linguagens de programação ALGOL e Pascal [7], ambas obsoletas [13]. Pascal, por exemplo, sequer está classificada entre as 50 linguagens de programação mais utilizadas no desenvolvimento de sistemas de software [13]. Além disso, Pascal apresenta alguns problemas em sua construção inerentes à linguagem, e.g., declaração das variáveis do bloco a priori. Assim, devido à origem sintática de Portugol, estudantes *não* terão uma transição natural ao aprender linguagens de programação do estado-da-prática, como C ANSI e Java.

Como uma alternativa ao Portugol, este artigo propõe PortuCol, uma pseudolinguagem de programação com instruções em português, assim como Portugol, porém inspirada em C ANSI. A escolha de C ANSI se justifica por pelo menos dois motivos: (i) juntamente com a linguagem Java são responsáveis por 35% de todos os sistemas de software em desenvolvimento [13]; e (ii) a linguagem C apresenta aplicações na educação, e.g., a plataforma de prototipagem eletrônica Arduino apresenta uma linguagem baseada em C [11] e pode ser usada em contextos de ensino e aprendizagem com o uso da robótica [2]. São apresentados estudos empíricos que atestam que (i) PortuCol é 27% mais similar a C ANSI do que o Portugol, o que proporciona uma transição mais natural para o estudante ao aprender linguagens do estado-da-prática e (ii) PortuCol é para C ANSI o que Portugol é para Pascal, o que retrata PortuCol como uma adaptação às linguagens do estado-da-prática. Além disso, este artigo descreve `PortuCol2C`, uma ferramenta que traduz código PortuCol para C ANSI, o que possibilita aos estudantes se familiarizem com a linguagem C ANSI através da observação do seu código PortuCol traduzido.

Este artigo está organizado como descrito a seguir. A Seção 2 apresenta Portugol, a pseudolinguagem mais adotada em disciplinas de programação do Brasil. A Seção 3 descreve o PortuCol, sua sintaxe e a ferramenta `PortuCol2C`. A Seção 4 reporta os resultados da avaliação de similaridade textual entre algoritmos implementados em PortuCol, C ANSI, Portugol e Pascal. Por fim, a Seção 5 conclui e propõe trabalhos futuros.

2. Portugol

O Portugol é a pseudolinguagem mais adotada em classes introdutórias à programação em IES no Brasil [12, 14]. Mais importante, a maioria dos livros utilizados como referência básica nas disciplinas de introdução à programação em universidades brasileiras possui explicações usando alguma notação de Portugol [12, 14]. Conclui-se que, o objetivo de Portugol é facilitar o aprendizado de Lógica de Programação e Algoritmos para alunos brasileiros não habituados com programação.

Portugol é o produto de uma simbiose do português e as linguagens de programação ALGOL e Pascal [7], ambas obsoletas [13]. Além disso, a linguagem Pascal tem problemas inerentes à linguagem. Um desses problemas é a declaração de variáveis que acontece antes do bloco de código ser iniciado, o que faz com que essas variáveis interfiram no bloco inteiro, contrariamente a somente interferir a partir do momento que são necessárias. Seu propósito é ser uma pseudolinguagem simples e com instruções em português para favorecer estudantes não familiares com a língua inglesa. Ao utilizar Portugol, o aluno não precisa se preocupar com a sintaxe e comandos em inglês, que uma linguagem de programação mais complexa possui, o que o permite focar sua concentração no *como* resolver o problema proposto.

Apesar de trazer benefícios no aprendizado de Lógica de Programação e Algoritmos, Portugol não provê, por definição, uma forma de executar o pseudocódigo escrito para que alunos possam testar seus algoritmos, o que se torna indispensável à medida que os algoritmos se tornam mais complexos e passam a requerer códigos mais elaborados. Esse problema ocorre pois o Portugol tem diversas variações em sintaxe e em comandos, dependendo do contexto de utilização. Isso faz com que as ferramentas não sejam padronizadas, i.e., suportando sintaxes e instruções diferentes [5, 7, 12, 10, 6].

Dentre as ferramentas existentes que apoiam o ensino utilizando Portugol podemos destacar: CIFluxProg [5], que disponibiliza um ambiente visual com fluxogramas, além de um ambiente textual com uma variação de Portugol; WEB-UNERJOL [7], que provê um ambiente *web* que permite o desenvolvimento e execução de algoritmos em Portugol através de um navegador; Portugol Studio [12], que é um ambiente de desenvolvimento integrado (IDE) para uma variante do Portugol que utiliza o compilador Portugol Núcleo; Portugol IDE [10], que provê um ambiente que possibilita desenvolver algoritmos em Portugol através do uso de fluxogramas como uma linguagem gráfica, além do tradicional modo textual; e, por fim, VisuAlg[6], que adota uma variante de Portugol bem parecida com Pascal, porém sem pontos-e-vírgulas para separar comandos. Os Algoritmos 1 e 2 ilustram tal similaridade apresentando um mesmo algoritmo escrito em Portugol e em Pascal, respectivamente.

Algoritmo 1. Portugol

```
1  algoritmo "helloWorld"
2  var
3      nome : caractere
4      i : inteiro
5  inicio
6      escreva("Digite seu nome: ")
7      leia(nome)
8      para i de 0 ate 2 passo 1 faca
9          escreva("Ola ", nome, "!")
10     fimpara
11 fimalgoritmo
```

Algoritmo 2. Pascal

```
1  Program helloWorld;
2  var
3      nome : string;
4      i : integer;
5  Begin
6      write('Digite seu nome: ');
7      readln(nome);
8      for i := 0 to 2 do
9      Begin
10         write('Ola ', nome, '!');
11     End;
12 End.
```

3. PortuCol

Este artigo propõe PortuCol, uma pseudolinguagem de programação com instruções em português, assim como Portugol, porém inspirada em C ANSI. O objetivo é fornecer uma ferramenta mais adequada ao ensino de Lógica de Programação e Algoritmos por meio

de uma pseudolinguagem cuja sintaxe seja inspirada em uma linguagem de programação do estado-da-prática (i.e., amplamente adotada por fábricas de software).

A Figura 1 ilustra o contexto de utilização de PortuCol. Estudantes desenvolvem o código em PortuCol e, automaticamente, a ferramenta `PortuCol2C` traduz PortuCol para C ANSI e executa o código correspondente. Isso possibilita que estudantes (i) executem seu pseudocódigo; (ii) se familiarizem com a linguagem C ANSI através da observação do seu código PortuCol traduzido; e (iii) tenham uma transição mais natural ao aprender linguagens do estado-da-prática como C ANSI e Java.

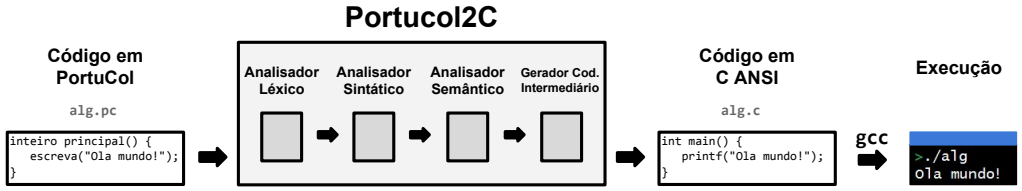

Figura 1. Utilizando PortuCol

3.1. Sintaxe

A definição da sintaxe de PortuCol considerou didática, facilidade de aprendizagem e similaridade com C ANSI, de forma que futuramente a transição para essa linguagem de programação seja mais natural para o aluno.

A Tabela 1 apresenta um subconjunto da equivalência de instruções entre PortuCol, C e Portugol. A tabela de equivalência de todas as instruções de PortuCol pode ser encontrada no Apêndice A. PortuCol possui instruções e palavras reservadas em português similarmente à Portugol e uma sintaxe similar à C ANSI. Em PortuCol, assim como em C ANSI, é necessário delimitar o escopo do código com os símbolos de abre e fecha chaves. Os operadores aritméticos e lógicos são como em C ANSI; exceções (e.g., operadores lógicos `e` e `ou`) ocorreram em virtude do caráter didático que a pseudolinguagem proposta tem como objetivo.

As estruturas básicas de desvio e repetição seguem, em sua maioria, a estrutura da linguagem de programação C ANSI. Contudo, uma das estruturas que foram modificadas com o intuito de apresentar um conceito de forma didática foi a estrutura de repetição `para` (`for`) que em PortuCol tem condições de parada fixas (menor ou igual, e maior ou igual). O objetivo dessa modificação foi de balancear a facilidade de aprendizado com o aspecto didático, presente também em Portugol, com a sintaxe de C ANSI que pode ser vista na estrutura do comando. Essa escolha também tem o objetivo de habituar o aluno a entender arranjos de tamanho n como estruturas que são acessadas a partir da posição 0 até a posição $n - 1$, o que acontece na maioria das linguagens de programação.

Usando o mesmo algoritmo descrito na Seção 2, os Algoritmos 3 e 4 apresentam dois exemplos do mesmo código em PortuCol e C ANSI, respectivamente.

Tabela 1. Equivalência de instruções PortuCol, C e Portugol

Construções	PortuCol	C	Portugol

Estrutura	`inteiro principal(){`
	`  ...`
	`}`
		`int main(){`
		`  ...`
		`}`
			`Algoritmo "nome"`
			`var`
			`  ...`
			`inicio`
			`  ...`
			`fimalgoritmo`

Variável tipo inteiro	`inteiro i;`	`int i;`
		`short int i;`
		`long int i;`
		`long long int i;`
			`i: inteiro`

Ler valor	`leia("%tipo", i);`	`scanf("%tipo", i);`	`leia(i)`
Operador e	`a e b`	`a && b`	`a e b`

Condicional caso	`escolha (variável) {`
	`  caso valor1:`
	`  ...`
	`  caso valor2, valor`
	`  ...`
	`  padrao:`
	`  ...`
	`}`
		`switch (variável) {`
		`  case valor1:`
		`  ...`
		`  break;`
		`  case valor2:`
		`  case valor3:`
		`  ...`
		`  break;`
		`  default:`
		`  ...`
		`  break;`
		`}`
			`escolha variável`
			`  caso valor1`
			`  ...`
			`  caso valor2, valor3`
			`  ...`
			`  outrocaso`
			`  ...`
			`fimescolha`

Condicional para	`para (variável = valor`
	`valorN ; P) {`
	`  ...`
	`}`
		`for (variável = valo`
		`variável <= valorN;`
		`variável += P) {`
		`  ...`
		`}`
			`para variável de valor1`
			`ate valorN passo P faca`
			`  ...`
			`fimpara`

Algoritmo 3. PortuCol

```
1  inteiro principal(){
2      texto nome;
3      escreva("Digite seu nome: ");
4      leia("%texto", nome);
5      int i;
6      para( i=0 ; 2 ; 1){
7          escreva("Ola %texto!", nome);
8      }
9  }
```

Algoritmo 4. C ANSI

```
1  int main(){
2      char nome[20];
3      printf("Digite seu nome: ");
4      scanf("%s", nome);
5      int i;
6      for( i=0 ; i<=2 ; i++){
7          printf("Ola %s!", nome);
8      }
9  }
```

3.2. Ferramenta `PortuCol2C`

Para que os estudantes possam executar seus códigos escritos em PortuCol, foi criado um protótipo de uma ferramenta, denominada `PortuCol2C`. Essa ferramenta realiza a tradução dos códigos em PortuCol para a linguagem C ANSI e, por conseguinte, compila e gera o arquivo executável correspondente. Dessa forma, `PortuCol2C` permite que estudantes (i) "executem" seu código PortuCol e (ii) observem qual seria o código C ANSI correspondente, o que já os familiariza com linguagens de programação reais.

Na atual versão da ferramenta, a execução de `PortuCol2C` é por linha de comando, conforme pode ser observado na Figura 2. Dado um código em PortuCol (`alg.pc`), a execução ferramenta (`portucol2c alg.pc`) gera o código C ANSI correspondente (`alg.c`) e o arquivo executável (`alg`).

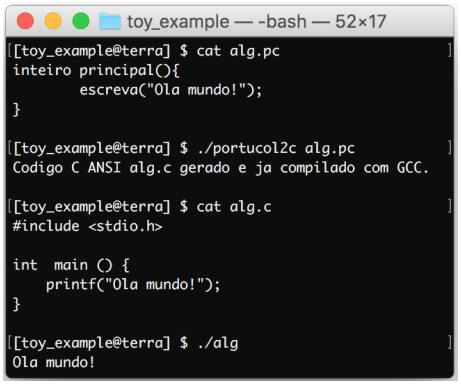

Figura 2. Execução do `PortuCol2C`

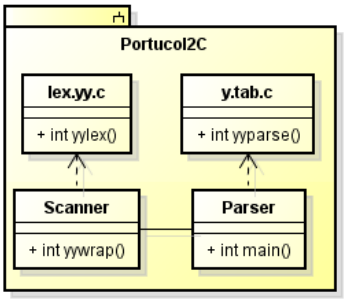

Figura 3. Arquitetura do `PortuCol2C`

O tradutor foi implementado usando *Lex* e *Yacc*,[1] que são ferramentas designadas para o desenvolvimento de compiladores e interpretadores [8]. O *Lex* realiza a análise léxica, que consiste na leitura dos caracteres de entrada e produção de uma sequência de *tokens*, enquanto o *Yacc* executa a análise sintática que consiste na obtenção de uma cadeia de *tokens* e verificação de quaisquer erros de sintaxe de uma forma inteligível [1]. Conforme ilustrado na Figura 3, a ferramenta segue então uma arquitetura com os seguintes dois módulos principais:

1. *Scanner*: Responsável por realizar a análise léxica do código-fonte escrito em PortuCol. Sua implementação consiste em definições de expressões regulares a serem convertidas em *tokens* tais como números, variáveis e palavras reservadas. Posteriormente, são definidas as regras para cada *token*, sendo que no caso do *Scanner* as regras estabelecidas consistem em retornar cada *token* para o *Parser*. A compilação do *Scanner* pelo *Lex* gera o arquivo `lex.yy.c`, o qual é utilizado para viabilizar a detecção dos *tokens* no código-fonte do PortuCol.
2. *Parser*: Responsável por realizar a análise sintática do código-fonte escrito em PortuCol. O *Parser* é constituído por regras formadas através dos *tokens* recebidos pelo *Scanner*, sendo que cada regra possui subrotinas associadas. No nosso caso, as subrotinas têm como finalidade imprimir, no arquivo de saída, a tradução para código C ANSI da sequência de *tokens* relacionado à cada regra. A compilação do *Parser* pelo *Yacc* gera o arquivo `y.tab.c`, o qual é utilizado para possibilitar a análise sintática dos *tokens*.

Os fontes dos analisadores léxico e sintático podem ser encontrados nos Apêndices B e C.

4. Avaliação

4.1. Questões de Pesquisa

Um estudo foi projetado para responder às seguintes questões de pesquisa:

QP #1 – PortuCol se aproxima à C ANSI mais do que Portugol?

QP #2 – PortuCol se aproxima à C ANSI mais do que Portugol se aproxima à Pascal?

[1] http://dinosaur.compilertools.net/

O objetivo da QP #1 é avaliar se PortuCol se aproxima mais de linguagens do estado-da-prática, confirmando que o objetivo foi alcançado. Já a QP #2 tem como objetivo verificar se PortuCol é mais próxima às linguagens do estado-da-prática do que Portugol é à Pascal.

4.2. *Dataset*[2]

Para efetivamente comparar a similaridade entre algoritmos nas linguagens PortuCol, C, Portugol e Pascal, foram especificados 30 algoritmos, os quais foram implementados em cada uma das linguagens. A Tabela 3 descreve sucintamente os algoritmos implementados e aponta as principais estruturas utilizadas.

Tabela 3. Descrição dos algoritmos

Algoritmo	Descrição	Principais estruturas utilizadas
`Alg_1`	Cadastro e acesso de produtos	Procedimentos, enquanto, para, caso, se
`Alg_2`	Equação quadrática	Tipo de dado real, se senao se
`Alg_3`	Idade por ano de nascimento	Tipo de dado arranjo, para
`Alg_4`	Bubble sort	Tipo de dado arranjo, para, se
`Alg_5`	Busca binária	Função, enquanto, se senao se
`Alg_6`	Busca em arranjo	Faca enquanto, se, senao
`Alg_7`	Calculadora (+, -, *)	Caso, tipo de dado texto
`Alg_8`	Média de notas	Leitura de entrada, se, senao
`Alg_9`	Soma e subtração de reais	Para, se, senao
`Alg_10`	Combinação	Escrita e leitura de dados, Para
`Alg_11`	Decomposição de inteiro	Para, faca enquanto, se
`Alg_12`	Aprovação de aluno por nota	Caso
`Alg_13`	Fatorial recursivo	Função recursiva, se senao se
`Alg_14`	Fibonacci iterativo	Função, para
`Alg_15`	Selection sort	Para, tipo de dado arranjo, se
`Alg_16`	Mostrar coluna de matriz	Matriz 4x4, para
`Alg_17`	Média de reais	Função, tipo de dados real
`Alg_18`	Inverter texto	Tipo de dados texto, para
`Alg_19`	Números palíndromos	Enquanto, se, senao
`Alg_20`	Primos até certo número	Para, enquanto, se
`Alg_21`	Aproximação de Pi	Para, operação de raiz quadrada
`Alg_22`	Potenciação de inteiros (xˆy)	Para
`Alg_23`	Comb sort	Enquanto, se, para
`Alg_24`	Soma de matrizes	Matrizes 2x2, para
`Alg_25`	Soma de algarismos	Enquanto, operação de resto de divisão
`Alg_26`	Classificação de texto por caso	Caso, comparação de texto
`Alg_27`	Validação de CPF	Para, se, conversão de texto para inteiro
`Alg_28`	Divisão de arranjos entre par e ímpar	Repita enquanto, para, se, senao
`Alg_29`	Aproximação de raiz quadrada	Se, enquanto, senao
`Alg_30`	Concatenação e comparação de texto	Se, senao, operações de concatenação e comparação de texto

4.3. Similaridade Textual

Para obter a similaridade entre os algoritmos implementados em PortuCol, C , Portugol e Pascal foi utilizada a similaridade de cosseno (*cosine similarity*) – técnica mais adotada em recuperação de informação e mineração de texto para comparação de documentos [9] – a qual mensura a similaridade de dois documentos com base no ângulo entre seus vetores representativos.[3]

Portanto, a similaridade de cosseno entre dois documentos d_1 e d_2 é computada a partir dos seus vetores representativos $\overrightarrow{V}(d_1)$ e $\overrightarrow{V}(d_2)$ conforme Equação 1.

$$sim(d_1, d_2) = \frac{\overrightarrow{V}(d_1) \cdot \overrightarrow{V}(d_2)}{|\overrightarrow{V}(d_1)||\overrightarrow{V}(d_1)|} \quad (1)$$

[2] *Dataset* publicamente disponível em: `http://www.dcc.ufla.br/~terra/papers/2016_wei_portucol`

[3] A similaridade varia de 0 (nada similares) a 1 (totalmente similares).

onde $\overrightarrow{V}(d_i)$ denota o vetor derivado de um documento d_i, com um componente no vetor para cada termo do dicionário. Assim, cada termo do documento é armazenado uma única vez no vetor e é associado a um peso. É importante enfatizar que os elementos presentes nos dois documentos estarão nos dois vetores representativos, apresentando peso zero no vetor representativo do documento em que o termo não está presente. O peso dos termos normalmente leva em consideração o $tf-idf_{t,d}$, onde tf é a frequência dos termos (*Term Frequency*), dada pela quantidade de vezes em que o termo aparece no documento em questão; idf é a frequência inversa do documento (*Inverse Document Frequency*), dada por uma função inversa que tem como objetivo diminuir o peso de termos que ocorrem com muita frequência em todos os documentos; t representa os termos e d o documento.[4] Enfim, a similaridade entre dois documentos consiste na razão entre (i) o produto escalar e (ii) o produto das normas euclidianas dos dois vetores representativos, que se trata do cosseno do ângulo entre os dois vetores.

4.4. Metodologia

Foram conduzidas as seguintes atividades para se responder às questões de pesquisa:

1. Definir uma particular versão de Portugol, devido à existência de variações do Portugol encontradas na literatura, onde diferentes ferramentas [5, 7, 12, 10, 6] possuem sua próprias variações de Portugol. Assim, foi escolhida a variante adotada pelo VisuAlg – um interpretador de Portugol desenvolvido por Souza [6] – pois é uma ferramenta estável e com o objetivo de facilitar o aprendizado de programação, características que foram verificadas em [4].
2. Construir um *dataset* contendo uma vasta gama de algoritmos implementados nas linguagens PortuCol, C, Portugol e Pascal (veja Seção 4.2).
3. Comparar a similaridade textual – utilizando *cosine similarity* descrito na Seção 4.3 – dos algoritmos implementados na linguagem PortuCol com C, Portugol com C e Portugol com Pascal. A decisão de utilizar um algoritmo de similaridade textual, contrariamente à um algoritmo de similaridade estrutural, se deve ao fato de que, como os algoritmos implementados nas diferentes linguagens tinham as mesmas estruturas nas mesmas ordens, as estruturas seriam idênticas na grande maioria dos casos. É importante mencionar que textos referentes à inclusão de bibliotecas (e.g. `#include <stdio.h>` e `uses crt`) foram retirados dos códigos C e Pascal.
4. Determinar estatisticamente a relação de similaridade de PortuCol com C e de Portugol com C, a fim de responder a QP #1.
5. Determinar estatisticamente a relação de similaridade de PortuCol com C e de Portugol com Pascal, a fim de responder a QP #2.

4.5. Resultados

A Tabela 4 reporta os resultados da similaridade textual dos algoritmos implementados na linguagem Portugol com C (coluna 1), PortuCol com C (coluna 2) e Portugol com Pascal (coluna 3).

[4]Particularmente neste artigo, foi considerada apenas a frequência dos termos, uma vez que diminuir o peso de termos frequentes não é desejável.

Tabela 4. Similaridade Textual

Algoritmo	Portugol/C	PortuCol/C	Portugol/Pascal
Alg_1	0.4004	0.6031	0.4365
Alg_2	0.7009	0.6824	0.7607
Alg_3	0.6348	0.7314	0.7291
Alg_4	0.7824	0.8713	0.7850
Alg_5	0.6803	0.7712	0.6591
Alg_6	0.5866	0.7072	0.6256
Alg_7	0.6674	0.6292	0.7227
Alg_8	0.6190	0.5930	0.6347
Alg_9	0.3036	0.7593	0.6543
Alg_10	0.7386	0.6616	0.8338
Alg_11	0.7262	0.8586	0.7523
Alg_12	0.2761	0.3639	0.5355
Alg_13	0.6385	0.6468	0.6156
Alg_14	0.6230	0.6863	0.5823
Alg_15	0.7190	0.8915	0.7118
Alg_16	0.5420	0.6057	0.5809
Alg_17	0.4449	0.4355	0.5536
Alg_18	0.6502	0.7662	0.7963
Alg_19	0.8511	0.8685	0.8348
Alg_20	0.3073	0.7611	0.7110
Alg_21	0.3267	0.7708	0.7780
Alg_22	0.6261	0.6199	0.6581
Alg_23	0.8580	0.9192	0.8247
Alg_24	0.0407	0.8187	0.9045
Alg_25	0.1079	0.8729	0.7621
Alg_26	0.6490	0.7286	0.7338
Alg_27	0.6202	0.8472	0.5964
Alg_28	0.5844	0.6647	0.6306
Alg_29	0.8454	0.8956	0.8604
Alg_30	0.4611	0.5484	0.4665
Média ($\overline{x}$)	**0.5671**	**0.7193**	**0,6910**
Dev. Pad. (s)	**0.2089**	**0.1361**	**0,1161**

QP #1 – *PortuCol se aproxima à C ANSI mais do que Portugol?*

Foram comparadas as similaridades textuais dos algoritmos implementados (i) em Portugol e em C e (ii) em PortuCol e em C, cujos resultados podem ser observados nas primeiras duas colunas da Tabela 4 e nas duas primeiras linhas da Figura 4.

O *dataset* disponível para avaliação apresenta os mesmos algoritmos implementados em diferentes linguagens (veja Seção 4.2). Isso permite que seja conduzida uma observação pareada dos resultados de similaridade textual. Portanto, considerando uma distribuição normal dos resultados, a distribuição probabilística *t-Student* foi utilizada no cálculo do intervalo de confiança para responder à QP #1, conforme Equação 2.[5]

O *dataset* utilizado possui trinta algoritmos ($n = 30$). A diferença da similaridade textual entre cada algoritmo Portugol/C e PortuCol/C (colunas 1 e 2 da Tabela 4) foi calculada. Assim, obteve-se a média aritmética ($\overline{x} = 0,1523$) e o desvio padrão ($s = 0,2158$) das diferenças. A confiança adotada para as amostras da distribuição de *t-Student* foi 99,9% com grau de liberdade de 29, onde $t_{[0.9995;29]}$ é igual a 3,659.

$$IC = \overline{x} \pm t_{[1-\frac{\alpha}{2};n-1]}(\frac{s}{\sqrt{n}}) = 0,1523 \pm 3,6590(\frac{0,2158}{\sqrt{30}}) = [0{,}0081\ ,\ 0{,}2964] \qquad (2)$$

Assim, o teste t resultou no intervalo de $[0{,}0081\ ,\ 0{,}2964]$, o qual *não* inclui zero. Isso indica que, com 99,9% de confiança, PortuCol se aproxima mais à C ANSI do que Portugol.

QP #2 – *PortuCol se aproxima à C ANSI mais do que Portugol se aproxima à Pascal?*

Similarmente à QP #1, foram comparadas as similaridades textuais dos algoritmos

[5]O teste t resulta em um intervalo de confiança. Caso o intervalo inclua zero, as amostras *não* são estatisticamente diferentes. Caso contrário, pode-se afirmar que as amostras *são* estatisticamente diferentes, o que, neste caso, indica que a similaridade em um par de linguagens é maior que na outra.

implementados (i) em PortuCol e em C e (ii) em Portugol e em Pascal, cujos resultados podem ser observados nas últimas duas colunas da Tabela 4 e nas duas últimas linhas da Figura 4. É importante mencionar que os resultados comparativos entre PortuCol e C já tinham sido previamente calculados na resposta à QP #1.

Considerando uma distribuição normal dos resultados, a distribuição probabilística *t-Student* novamente foi utilizada no cálculo do intervalo de confiança para responder à QP #2, conforme Equação 3. Similarmente à *QP #1*, a confiança adotada para as amostras da distribuição de *t-Student* foi 99,9% com grau de liberdade de 29.

$$IC = \overline{x} \pm t_{[1-\frac{\alpha}{2};n-1]}(\frac{s}{\sqrt{n}}) = 0,0283 \pm 3.659(\frac{0.1006}{\sqrt{30}}) = [-0{,}0389\ ,\ 0{,}0955] \quad (3)$$

Assim, o teste t resultou no intervalo de $[-0{,}0389\ ,\ 0{,}0955]$. Como o intervalo de confiança inclui zero, podemos considerar que o PortuCol se aproxima à C ANSI da mesma maneira que Portugol se aproxima à Pascal. A Figura 4 representa visualmente o intervalo de confiança para a média das similaridades textuais de cada par de linguagens.

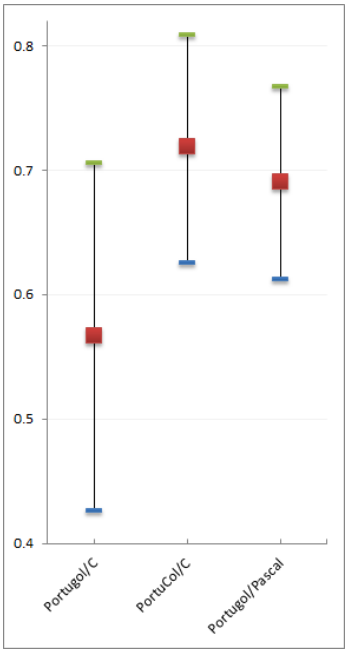

Figura 4. Intervalo de confiança

4.6. Ameaças à Validade

É importante mencionar pelo menos uma ameaça à validade de construção. A avaliação de similaridade entre PortuCol, C e Portugol analisou 30 algoritmos. A escolha desses algoritmos pode ter favorecido ou desfavorecido PortuCol na análise de similaridade. No entanto, foram escolhidos os 30 sistemas com as mais diversas estruturas presentes em sua construção e de forma completamente aleatória.

5. Considerações Finais

O uso de pseudolinguagens no ensino de Lógica de Programação e Algoritmos é uma estratégia muito utilizada em IES. Embora C ANSI e Java sejam as linguagens de programa-

ção mais utilizadas no desenvolvimento de sistemas de software [13], a pseudolinguagem mais utilizada no Brasil é o Portugol [12, 14], que é inspirada no obsoleto Pascal.

Diante desse cenário, este artigo propôs PortuCol, uma pseudolinguagem de programação com instruções em português assim como Portugol, porém inspirada em C ANSI. Como contribuição acadêmica, uma análise estatística da similaridade textual demostrou que: (i) a pseudolinguagem PortuCol é mais similar à C ANSI do que Portugol, o que proporciona uma transição mais natural ao estudante ao aprender linguagens do estado-da-prática como C ANSI e Java; e (ii) PortuCol é para C ANSI o que Portugol é para Pascal, o que retrata apenas uma adaptação da pseudolinguagem às linguagens do estado-da-prática. Como contribuição prática, este artigo também apresenta `PortuCol2C`, uma ferramenta que executa código PortuCol através da prévia tradução para código C ANSI, o que possibilita aos estudantes se familiarizem com a linguagem C ANSI através da observação do seu código PortuCol traduzido.

O *dataset* dos algoritmos utilizados na avaliação e a ferramenta `PortuCol2C` (binário e código fonte) estão publicamente disponíveis em: `http://www.dcc.ufla.br/~terra/papers/2016_wei_portucol`

Como trabalhos futuros, pretende-se (i) avaliar o uso da pseudolinguagem PortuCol no ensino de Lógica de Programação e Algoritmos; (ii) melhorar a ferramenta `PortuCol2C` com um ambiente de desenvolvimento integrado (IDE) e um depurador; e (iii) realizar a verificação de similaridade textual na formalização das gramáticas das linguagens, potencialmente no Formalismo de Backus-Naur (BNF).

Agradecimentos

Este trabalho foi apoiado pela FAPEMIG, CAPES e CNPq.

Referências

[1] Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullman. *Compiladores: Princıpios, técnicas e ferramentas*. LTC, 2 edition, 2008.

[2] Rafael Machado Alves, Armando Luiz Costa da Silva, Marcos de Castro Pinto, Fabio Ferrentini Sampaio, and Marcos da Fonseca Elia. Uso do hardware livre arduino em ambientes de ensino-aprendizagem. *Jornada de Atualização em Informática na Educação*, 1(1):162–187, 2013.

[3] Ricardo Luiz B. L. Campos. Lógica de programação: Há como melhorar o aprendizado fugindo dos padrões estabelecidos nos livros didáticos e adotados pela maioria dos docentes? In *XVII Congresso Iberoamericano de Educación Superior em Computación (CIESC)*, pages 22–25, 2009.

[4] Sormany Silva Dantas, Luiz Augusto de Macedo Morais, Rivanilson da Silva Rodrigues, Rodrigo Alves Costa, et al. Avaliação de um software educacional de apoio à aprendizagem de programação: Visualg. In *VII CONNEPI-Congresso Norte Nordeste de Pesquisa e Inovação*, 2012.

[5] Rafael de Santiago and Rudimar Luís Scaranto Dazzi. Ferramenta de apoio ao ensino de algoritmos. In *XIII Seminário de Computação (SEMINCO)*, pages 1–8, 2004.

[6] Cláudio Morgado de Souza. VisuAlg - ferramenta de apoio ao ensino de programação. *TECCEN*, 2(2):1–9, 2009.

[7] Mauri Ferrandin and Simone Lilian Stephani. Ferramenta para o ensino de programação via internet. In *I Congresso Sul Catarinense de Computação (SULCOMP)*, pages 1–8, 2005.

[8] John R. Levine, Tony Mason, and Doug Brown. *Lex & Yacc*. O'Reilly, 2 edition, 1992.

[9] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*. Cambridge University Press, 2008.

[10] António Manso, Luís Oliveira, and Célio Gonçalo Marques. Ambiente de aprendizagem de algoritmos – Portugol IDE. In *VI Conferência Internacional de TIC na Educação*, pages 969–983, 2009.

[11] Michael McRoberts. Arduino básico. *Editora Novatec*, 344755160, 2011.

[12] Luiz F. Noschang, Fillipi Pelz, Elieser A. de Jesus, and André L. A. Raabe. Portugol Studio: Uma IDE para iniciantes em programação. In *XXII Workshop sobre Educação em Computação (WEI)*, pages 1287–1296, 2014.

[13] TIOBE. Programming languages index. `http://www.tiobe.com/tiobe_index`, March 2016.

[14] Adriana Salvador Zanini and André Luís Alice Raabe. Análise dos enunciados utilizados nos problemas de programação introdutória em cursos de ciência da computação no Brasil. In *XX Workshop sobre Educação em Computação (WEI)*, 2012.

A. Equivalência de instruções entre PortuCol, C e Portugol

A Tabela 5 apresenta a equivalência de instruções entre Portucol, C e Portugol.

B. Analisador Léxico

A Listagem 5 apresenta o código do analisador léxico, parte do módulo *Scanner*.

Listagem 5. Analisador léxico

```
1  %{
2  /*--------------------------- Definitions ---------------------------*/
3  #include <stdio.h>
4  #include "y.tab.h"
5
6
7  /*Prototipo*/
8  void installID(){
9
10 }
11
12 %}
13
14 abrechave               "{"
15 fechachave               "}"
16 abrecolchete            "["
17 fechacolchete           "]"
18 abreparentese           "("
19 fechaparentese          ")"
20 underline               "_"
21 doispontos              ":"
22 aspasimples             "'"
23 exclamacao              "!"
24 string                  \"[^"\n]*["\n]
25 caractere               '([^'\\\n]|\\.)'
26 raizquadrada            "raizquadrada"
27 concatena               "concatena"
28 potencia                "potencia"
29 maiuscula               "maiuscula"
30 tamanhotexto            "tamanhotexto"
31 e                       "e"
32 ou                      "ou"
33 somasub                 ("+"|"-")
34 digito                  [0-9]
35 numint                  {digito}+
36 letra                   [a-z|A-Z]
37 se                      "se"
38 senao                   "senao"
39 faca                    "faca"
40 enquanto                "enquanto"
41 para                    "para"
42 retorne                 "retorne"
43 inteiro                 "inteiro"
44 real                    "real"
45 texto                   "texto"
46 logico                  "logico"
47 verdadeiro              "verdadeiro"
48 falso                   "falso"
49 semretorno              "semretorno"
50 escolha                 "escolha"
51 caso                    "caso"
52 padrao                  "padrao"
53 interrompa              "interrompa"
54 principal               "principal"
55 escreva                 "escreva"
56 leia                    "leia"
57 porreal                 "%real"
58 portexto                "%texto"
59 porinteiro              "%inteiro"
60 id                      {letra}(({letra}|{digito})|({letra}|{digito}|{underline}))*
61 numreal                 {digito}+(\.{digito}+)?([E][+-]?{digito}+)?
62 multdiv                 ("*"|"/"|"%")
```

Tabela 5. Equivalência de instruções PortuCol, C e Portugol

Construções	PortuCol	C	Portugol

Estrutura	`inteiro principal(){`
	`  ...`
	`}`
		`int main(){`
		`  ...`
		`}`
			`Algoritmo "nome"`
			`var`
			`  ...`
			`inicio`
			`  ...`
			`fimalgoritmo`

Variável tipo inteiro	`inteiro i;`	`int i;`
		`short int i;`
		`long int i;`
		`long long int i;`
			`i: inteiro`

Variável tipo real	`real d;`	`float d;`
		`double d;`
		`long double d;`
			`d: real`

Variável tipo lógico	`logico b;`	`bool b; (C99)`	`b: logico`
Variável tipo caractere	`texto s;`	`char s;`	`s: caractere`
Variável tipo *string*	`texto s;`	`char s[];`	`s: caractere`
Variável tipo vetor	`tipo arranjo[linhas];`	`tipo arranjo[linhas];`	`v: arranjo[linha1..linhaN] de tipo`
Variável tipo matriz	`tipo arranjo[linhas][colunas];`	`tipo arranjo[linhas][colunas];`	`v: arranjo[linha1..linhaN, coluna1..colunaN] de tipo`
Ler valor	`leia("%tipo", i);`	`scanf("%tipo", i);`	`leia(i)`
Escrever	`escreva("...");`	`printf("...");`	`escreva("...")`
Escrever com quebra de linha	`escreva("...\n");`	`printf("...\n");`	`escreval("...")`
Escrever com parâmetros	`escreva("... %tipo ...` `%tipo ...", a, b);`	`printf("... %tipo ...` `%tipo ...", a, b);`	`escreval("...", a, "...",` `b, "...")`
Operação de atribuição	`a = b;`	`a = b;`	`a <- b` `a := b`
Operação de adição	`a + b;`	`a + b;`	`a + b`
Operação de subtração	`a - b;`	`a - b;`	`a - b`
Operação de multiplicação	`a * b;`	`a * b;`	`a * b`
Operação de divisão	`a / b;`	`a / b;`	`a / b`
Operação de divisão inteira	`a / b;`	`a / b;`	`a div b`
Operação de resto da divisão	`a % b;`	`a % b;`	`a % b` `a mod b`
Operação de potenciação	`potencia(a, b);`	`pow(a, b)  <math.h>`	`a ^ b`
Operação de raiz quadrada	`raizquadrada(a);`	`sqrt(a);  <math.h>`	`RaizQ(a)`

Operação de concatenação	`concatena("texto","texto");`
	`concatena("texto", a);`
	`concatena(a, "texto");`
		`strcat("texto", "texto");`
		`strcat("texto", a);`
		`strcat(a, "texto");`
			`"texto" + "texto"`
			`"texto" + a (string)`
			`a (string) + "texto"`

Operação de tamanho do texto	`tamanhotexto(a);`	`strlen(a);  <string.h>`	`compr(a)`
Operador menor	`a < b;`	`a < b;`	`a < b`
Operador menor ou igual	`a <= b;`	`a <= b;`	`a <= b`
Operador maior	`a > b;`	`a > b;`	`a > b`
Operador maior ou igual	`a >= b;`	`a >= b;`	`a >= b`
Operador igual	`a == b;`	`a == b;`	`a = b`
Operador diferente	`a != b;`	`a != b;`	`a <> b`
Operador e	`a e b`	`a && b`	`a e b`
Operador ou	`a ou b`	`a || b`	`a ou b`
Operador de negação	`!a`	`!a`	`nao a`

Condicional se	`se (condição) {`
	`  ...`
	`} senao {`
	`  ...`
	`}`
		`if (condição) {`
		`  ...`
		`} else {`
		`  ...`
		`}`
			`se condição entao`
			`  ...`
			`senao`
			`  ...`
			`fimse`

Condicional aninhado	`se (condição){`
	`  ...`
	`} senao se (condição){`
	`  ...`
	`} senao {`
	`  ...`
	`}`
		`if (condição){`
		`  ...`
		`} else if (condição){`
		`  ...`
		`} else {`
		`  ...`
		`}`
			`se condição entao`
			`  ...`
			`senao`
			`  se condicao entao`
			`  ...`
			`  senao`
			`  ...`
			`  fimse`
			`fimse`

Condicional caso	`escolha (variável) {`
	`  caso valor1:`
	`  ...`
	`  caso valor2, valor3:`
	`  ...`
	`  padrao:`
	`  ...`
	`}`
		`switch (variável) {`
		`  case valor1:`
		`  ...`
		`  break;`
		`  case valor2:`
		`  case valor3:`
		`  ...`
		`  break;`
		`  default:`
		`  ...`
		`  break;`
		`}`
			`escolha variável`
			`  caso valor1`
			`  ...`
			`  caso valor2, valor3`
			`  ...`
			`  outrocaso`
			`  ...`
			`fimescolha`

Condicional enquanto	`enquanto (condição) {`
	`  ...`
	`}`
		`while (condição) {`
		`  ...`
		`}`
			`enquanto condição faca`
			`  ...`
			`Fimenquanto`

Condicional para	`para (variável = valor1 ;`
	`valorN ; P) {`
	`  ...`
	`}`
		`for (variável = valor1;`
		`variável <= valorN;`
		`variável += P) {`
		`  ...`
		`}`
			`para variável de valor1`
			`ate valorN passo P faca`
			`  ...`
			`fimpara`

Comando de interrupção	`interrompa;`	`break;`	`interrompa`

Criar procedimento	`semretorno nome(tipo`
	`parâmetro, ...) {`
	`  ...`
	`}`
		`void nome(tipo parâmetro,`
		`...) {`
		`  ...`
		`}`
			`procedimento nome(`
			`parâmetro: tipo; ...)`
			`var`
			`  ...`
			`inicio`
			`  ...`
			`fimprocedimento`

Criar função	`tipo nome(tipo parâmetro,`
	`...) {`
	`  ...`
	`  retorne valor;`
	`}`
		`tipo nome(tipo parâmetro,`
		`...) {`
		`  ...`
		`  return valor;`
		`}`
			`funcao nome(parâmetro:`
			`tipo; ...): tipo`
			`var`
			`  ...`
			`inicio`
			`  ...`
			`retorne valor`
			`fimfuncao`

```
relacional              ("<"|">"|"=="|">="|"<="|"!=")
atribuicao              "="
comentario              "/*"((\*+[^/*])|([^*]))*\**"*/"
virgula                 ","
pontoevirgula           ";"
delim                   [ \t\n]
ws                      {delim}+

other                   .

%%
%{
/*------------------------------ Rules -----------------------------*/
%}
{ws}                { /*sem acao, sem retorno */ }
{comentario}        {
                        int i;
                        for(i=0;i<yyleng;i++){
                            if(yytext[i]=='\n');
                        }
                    }
{string}            { return STRING; }
{caractere}         { return CARACTERE; }
{abrechave}         { return ABRECHAVE; }
{fechachave}        { return FECHACHAVE; }
{abrecolchete}      { return ABRECOLCHETE; }
{fechacolchete}     { return FECHACOLCHETE; }
{abreparentese}     { return ABREPARENTESE; }
{fechaparentese}    { return FECHAPARENTESE; }
{doispontos}        { return DOISPONTOS; }
{aspasimples}       { return ASPASIMPLES; }
{exclamacao}        { return EXCLAMACAO; }
{raizquadrada}      { return RAIZQUADRADA; }
{concatena}         { return CONCATENA; }
{potencia}          { return POTENCIA; }
{maiuscula}         { return MAIUSCULA; }
{tamanhotexto}      { return TAMANHOTEXTO; }
{e}                 { return E; }
{ou}                { return OU; }
{real}              { return REAL; }
{texto}             { return TEXTO; }
{inteiro}           { return INTEIRO; }
{logico}            { return LOGICO; }
{verdadeiro}        { return VERDADEIRO; }
{falso}             { return FALSO; }
{porinteiro}        { return PORINTEIRO; }
{porreal}           { return PORREAL; }
{portexto}          { return PORTEXTO; }
{semretorno}        { return SEMRETORNO; }
{principal}         { return PRINCIPAL; }
{escreva}           { return ESCREVA; }
{leia}              { return LEIA; }
{escolha}           { return ESCOLHA; }
{caso}              { return CASO; }
{padrao}            { return PADRAO; }
{interrompa}        { return INTERROMPA; }
{se}                { return SE; }
{senao}             { return SENAO; }
{faca}              { return FACA; }
{enquanto}          { return ENQUANTO; }
{para}              { return PARA; }
{retorne}           { return RETORNE; }
{id}                { installID();
                      return ID; }
{atribuicao}        { return ATRIBUICAO;}
{multdiv}           { return MULTDIV; }
{somasub}           { return SOMASUB; }
{numint}            { return NUMINT; }
{numreal}           { return NUMREAL; }
{virgula}           { return VIRGULA; }
{pontoevirgula}     { return PONTOEVIRGULA; }
{relacional}        { return RELACIONAL; }

{other}             { /*sem acao, sem retorno */ }
%%
/*------------------------ User subrotines -------------------------*/

int yywrap() {
    return 1;
}
```

C. Analisador Sintático

A Listagem 6 apresenta o código do analisador sintático, parte do módulo *Parser*.

Listagem 6. Analisador sintático

```
%{
    #include <stdio.h>
    #include <stdlib.h>
    #include <string.h>
    #include <stdbool.h>
    #include "replace.h"
    #define YYSTYPE char*
    int yylex(void);
    void yyerror(char *);
    extern FILE *yyin, *yyout;
    extern char* yytext;

    struct table {
        char tipo[10];
        char nome[30];
        char vet[100];
        int flag_param;
        int flag_char;
        int flag_string;
    };

    struct table tabela[300];
    int current_tab_tipo = 0, current_tab_nome = 0;
    int ident = 0;
%}
```

```
%token ABRECHAVE FECHACHAVE ABRECOLCHETE FECHACOLCHETE ABREPARENTESE FECHAPARENTESE
%token DOISPONTOS ASPASIMPLES EXCLAMACAO RAIZQUADRADA CONCATENA POTENCIA MAIUSCULA TAMANHOTEXTO E OU SOMASUB
      NUMINT NUMREAL
%token SE SENAO FACA ENQUANTO PARA RETORNE INTEIRO REAL TEXTO LOGICO VERDADEIRO FALSO SEMRETORNO ESCOLHA CASO
      PADRAO INTERROMPA PRINCIPAL LEIA ESCREVA PORREAL PORTEXTO PORINTEIRO ID
%token MULTDIV RELACIONAL ATRIBUICAO COMENTARIO VIRGULA PONTOEVIRGULA STRING CARACTERE

%nonassoc LOWER_THAN_SENAO
%nonassoc SENAO
%left  SOMASUB
%left  MULTDIV
%%
programa    : declaracao_lista { fprintf(yyout,"%s", $1); }
            ;
declaracao_lista : declaracao
                 {
                     char aux[10000];
                     strcpy(aux, $1);
                     int i;
                     char *result;
                     for(i = 0; i<current_tab_tipo ; i++){
                         if(tabela[i].flag_param == 1){
                             char temp_par[25];
                             char result_par[50];
                             char temp_vir[25];
                             char result_vir[20];

                             strcpy(temp_par, "(");
                             strcat(temp_par, tabela[i].tipo);
                             strcpy(result_par, temp_par);
                             if(strcmp(tabela[i].tipo, "char ") == 0){
                                 strcat(result_par, tabela[i].nome);
                                 strcat(result_par, tabela[i].vet);
                             } else {
                                 strcat(result_par, "*");
                                 strcat(result_par, tabela[i].nome);
                             }
                             strcat(temp_par, tabela[i].nome);

                             strcpy(temp_vir, ", ");
                             strcat(temp_vir, tabela[i].tipo);
                             strcpy(result_vir, temp_vir);
                             if(strcmp(tabela[i].tipo, "char ") == 0){
                                 strcat(result_vir, tabela[i].nome);
                                 strcat(result_vir, tabela[i].vet);
                             } else {
                                 strcat(result_vir, "*");
                                 strcat(result_vir, tabela[i].nome);
                             }
                             strcat(temp_vir, tabela[i].nome);

                             result = repl_str(aux, temp_par, result_par);
                             result = repl_str(result, temp_vir, result_vir);

                             char temp[100];
                             strcpy(temp, result_vir);
                             strcat(temp, tabela[i].vet);
                             result = repl_str(result, temp, result_vir);

                             char temp2[100];
                             strcpy(temp2, result_par);
                             strcat(temp2, tabela[i].vet);
                             result = repl_str(result, temp2, result_par);


                             strcpy(aux, result);
                         }
                         if(tabela[i].flag_char == 1){
                             char temp_decl[40];
                             char result_decl[40];
                             char temp_gets[70];
                             char result_gets[90];

                             strcpy(temp_decl, tabela[i].nome);
                             strcat(temp_decl, "[50]");
                             strcpy(result_decl, tabela[i].nome);

                             strcpy(temp_gets, "fflush(stdin);\ngets(");
                             strcat(temp_gets, tabela[i].nome);
                             strcpy(result_gets, "scanf(\" %c\", &");
                             strcat(result_gets, tabela[i].nome);

                             result = repl_str(aux, temp_decl, result_decl);
                             result = repl_str(result, temp_gets, result_gets);
                             strcpy(aux, result);
                         }
                     }
                     $$ = strdup(aux);
                     free(result);
                 }
    | declaracao declaracao_lista
                 {
                     char aux[10000];
                     strcpy(aux, $1);
                     strcat(aux, $2);
                     int i;
                     char *result;
                     for(i = 0; i<current_tab_tipo ; i++){
                         if(tabela[i].flag_param == 1){
                             char temp_par[25];
                             char result_par[50];
                             char temp_vir[25];
                             char result_vir[50];

                             strcpy(temp_par, "(");
                             strcat(temp_par, tabela[i].tipo);
                             strcpy(result_par, temp_par);
                             if(strcmp(tabela[i].tipo, "char ") == 0){
                                 strcat(result_par, tabela[i].nome);
                                 strcat(result_par, tabela[i].vet);
                             } else {
                                 strcat(result_par, "*");
                                 strcat(result_par, tabela[i].nome);
                             }
                             strcat(temp_par, tabela[i].nome);

                             strcpy(temp_vir, ", ");
                             strcat(temp_vir, tabela[i].tipo);
                             strcpy(result_vir, temp_vir);
                             if(strcmp(tabela[i].tipo, "char ") == 0){
                                 strcat(result_vir, tabela[i].nome);
                                 strcat(result_vir, tabela[i].vet);
                             } else {
                                 strcat(result_vir, "*");
                                 strcat(result_vir, tabela[i].nome);
```

```
                                }
                                strcat(temp_vir, tabela[i].nome);

                                result = repl_str(aux, temp_par, result_par);
                                result = repl_str(result, temp_vir, result_vir);

                                char temp[100];
                                strcpy(temp, result_vir);
                                strcat(temp, tabela[i].vet);
                                result = repl_str(result, temp, result_vir);

                                char temp2[100];
                                strcpy(temp2, result_par);
                                strcat(temp2, tabela[i].vet);
                                result = repl_str(result, temp2, result_par);


                                strcpy(aux, result);
                            }
                            if(tabela[i].flag_char == 1){
                                char temp_decl[40];
                                char result_decl[40];
                                char temp_gets[70];
                                char result_gets[90];

                                strcpy(temp_decl, tabela[i].nome);
                                strcat(temp_decl, "[50]");
                                strcpy(result_decl, tabela[i].nome);

                                strcpy(temp_gets, "fflush(stdin);\ngets(");
                                strcat(temp_gets, tabela[i].nome);
                                strcpy(result_gets, "scanf(\" %c\", &");
                                strcat(result_gets, tabela[i].nome);

                                result = repl_str(aux, temp_decl, result_decl);
                                result = repl_str(result, temp_gets, result_gets);
                                strcpy(aux, result);
                            }
                        }
                        $$ = strdup(aux);
                        free(result);
                    }
                    ;
declaracao  : var_declaracao
                | fun_declaracao
                ;
var_declaracao  : tipo_especificador var_declaracao_meio
                    {
                        char aux[100];
                        strcpy(aux, $1);
                        strcat(aux, $2);
                        while(current_tab_tipo < current_tab_nome){
                            strcpy(tabela[current_tab_tipo].tipo, $1);
                            current_tab_tipo++;
                        }
                        $$ = strdup(aux);
                    }
                | TEXTO decl_texto
                    {
                        char aux[100];
                        strcpy(aux, "char ");
                        strcat(aux, $2);
                        $$ = strdup(aux);
                    }
                ;
decl_texto  : id decl_texto_vet decl_texto_fat
                    {
                            char aux[100];
                            strcpy(aux, $1);
                            strcat(aux, $2);
                            strcpy(tabela[current_tab_nome].nome, $1);
                            strcat(aux, "[50]");
                            strcpy(tabela[current_tab_tipo].tipo, "char ");
                            strcpy(tabela[current_tab_nome].vet, $2);
                            strcat(tabela[current_tab_nome].vet, "[50]");
                            tabela[current_tab_nome].flag_string = 1;
                            current_tab_tipo++;
                            current_tab_nome++;
                            strcat(aux, $3);
                            $$ = strdup(aux);
                    }
                ;
decl_texto_vet  : ABRECOLCHETE numint FECHACOLCHETE ATRIBUICAO var_vet_atr
                    {
                        char aux[200];
                        strcpy(aux, "[");
                        strcat(aux, $2);
                        strcat(aux, "] = ");
                        strcat(aux, $5);
                        $$ = strdup(aux);
                    }
                | ABRECOLCHETE numint FECHACOLCHETE decl_texto_vet
                    {
                        char aux[100];
                        strcpy(aux, "[");
                        strcat(aux, $2);
                        strcat(aux, "]");
                        strcat(aux, $4);
                        $$ = strdup(aux);
                    }
                | /*vazio*/     { $$ = strdup(""); }
                ;
decl_texto_fat  : VIRGULA decl_texto
                    {
                        char aux[100];
                        strcpy(aux, ", ");
                        strcat(aux, $2);
                        $$ = strdup(aux);
                    }
                | /*vazio*/     { $$ = strdup(""); }
                ;
var_declaracao_meio : id var_declaracao_vet var_declaracao_fat
                        {
                            char aux[100];
                            strcpy(aux, $1);
                            strcat(aux, $2);
                            strcpy(tabela[current_tab_nome].nome, $1);
                            strcpy(tabela[current_tab_nome].vet, $2);
                            current_tab_nome++;
                            strcat(aux, $3);
                            $$ = strdup(aux);
                        }
                    ;
var_declaracao_vet  : ABRECOLCHETE numint FECHACOLCHETE ATRIBUICAO var_vet_atr
```

```
                        {
                            char aux[200];
                            strcpy(aux, "[");
                            strcat(aux, $2);
                            strcat(aux, "] = ");
                            strcat(aux, $5);
                            $$ = strdup(aux);
                        }
                | ABRECOLCHETE numint FECHACOLCHETE var_declaracao_vet
                    {
                        char aux[100];
                        strcpy(aux, "[");
                        strcat(aux, $2);
                        strcat(aux, "]");
                        strcat(aux, $4);
                        $$ = strdup(aux);
                    }
                | ATRIBUICAO expressao
                    {
                        char aux[100];
                        strcpy(aux, " = ");
                        strcat(aux, $2);
                        $$ = strdup(aux);
                    }
                | /*vazio*/     { $$ = strdup(""); }
                ;
var_declaracao_fat  : VIRGULA var_declaracao_meio
                        {
                            char aux[100];
                            strcpy(aux, ", ");
                            strcat(aux, $2);
                            $$ = strdup(aux);
                        }
                | PONTOEVIRGULA     { $$ = strdup(";\n"); }
                ;
var_vet_atr : ABRECHAVE var_vet_dentro FECHACHAVE
                {
                    char aux[100];
                    strcpy(aux, "{");
                    strcat(aux, $2);
                    strcat(aux, "}");
                    $$ = strdup(aux);
                }
                ;
var_vet_dentro  : var_vet_atr VIRGULA var_vet_atr
                    {
                        char aux[100];
                        strcpy(aux, $1);
                        strcat(aux, ", ");
                        strcat(aux, $3);
                        $$ = strdup(aux);
                    }
                | conteudo_vetor
                ;
conteudo_vetor  : numint vetor_int
                    {
                        char aux[100];
                        strcpy(aux, $1);
                        strcat(aux, $2);
                        $$ = strdup(aux);
                    }
                | numreal vetor_real
                    {
                        char aux[100];
                        strcpy(aux, $1);
                        strcat(aux, $2);
                        $$ = strdup(aux);
                    }
                | string vetor_string
                    {
                        char aux[100];
                        strcpy(aux, $1);
                        strcat(aux, $2);
                        $$ = strdup(aux);
                    }
                | caractere vetor_caractere
                    {
                        char aux[100];
                        strcpy(aux, $1);
                        strcat(aux, $2);
                        $$ = strdup(aux);
                    }
                ;
vetor_int   : VIRGULA numint vetor_int
                {
                    char aux[100];
                    strcpy(aux, ", ");
                    strcat(aux, $2);
                    strcat(aux, $3);
                    $$ = strdup(aux);
                }
                | /*vazio*/     { $$ = strdup(""); }
                ;
vetor_real  : VIRGULA numreal vetor_real
                {
                    char aux[100];
                    strcpy(aux, ", ");
                    strcat(aux, $2);
                    strcat(aux, $3);
                    $$ = strdup(aux);
                }
                | /*vazio*/     { $$ = strdup(""); }
                ;
vetor_string    : VIRGULA string vetor_string
                    {
                        char aux[100];
                        strcpy(aux, ", ");
                        strcat(aux, $2);
                        strcat(aux, $3);
                        $$ = strdup(aux);
                    }
                | /*vazio*/     { $$ = strdup(""); }
                ;
vetor_caractere : VIRGULA caractere vetor_caractere
                    {
                        char aux[100];
                        strcpy(aux, ", ");
                        strcat(aux, $2);
                        strcat(aux, $3);
                        $$ = strdup(aux);
                    }
                | /*vazio*/     { $$ = strdup(""); }
                ;
tipo_especificador  : INTEIRO       { $$ = "int "; }
```

```
                | REAL      { $$ = "float "; }
                | SEMRETORNO        { $$ = "void "; }
                | LOGICO        { $$ = "bool "; }
                  ;
id          : ID    { $$ = strdup(yytext); }
                  ;
numint      : NUMINT        { $$ = strdup(yytext); }
                  ;
numreal     : NUMREAL       { $$ = strdup(yytext); }
                  ;
string      : STRING        { $$ = strdup(yytext); }
                  ;
caractere   : CARACTERE     { $$ = strdup(yytext); }
                  ;
fun_declaracao  : tipo_especificador id ABREPARENTESE params FECHAPARENTESE composto_decl
                    {
                        char aux[5000];
                        strcpy(aux, $1);
                        strcat(aux, $2);
                        strcat(aux, " (");
                        strcat(aux, $4);
                        strcat(aux, ")");
                        strcat(aux, $6);
                        $$ = strdup(aux);
                    }
                | tipo_especificador PRINCIPAL ABREPARENTESE params FECHAPARENTESE composto_decl
                    {
                        char aux[5000];
                        strcpy(aux, $1);
                        strcat(aux, " main (");
                        strcat(aux, $4);
                        strcat(aux, ")");
                        strcat(aux, $6);
                        $$ = strdup(aux);
                    }
                  ;
params              : param_lista
                | /*vazio*/         { $$ = strdup(""); }
                  ;
param_lista     : param param_lista_fat
                    {
                        char aux[100];
                        strcpy(aux, $1);
                        strcat(aux, $2);
                        $$ = strdup(aux);
                    }
                      ;
param_lista_fat     : VIRGULA param_lista
                        {
                            char aux[100];
                            strcpy(aux, ", ");
                            strcat(aux, $2);
                            $$ = strdup(aux);
                        }
                | /*vazio*/         { $$ = strdup(""); }
                  ;
param   : tipo_especificador id
            {
                char aux[100];
                strcpy(aux, $1);
                strcat(aux, $2);
                $$ = strdup(aux);
            }
                | TEXTO id
                    {
                        char aux[100];
                        strcpy(aux, "char ");
                        strcat(aux, $2);
                        $$ = strdup(aux);
                    }
                  ;
composto_decl   : ABRECHAVE dentro_funcao FECHACHAVE
                    {
                        char aux[100];
                        strcpy(aux, "{ \n");
                        strcat(aux, $2);
                        strcat(aux, "}\n");
                        $$ = strdup(aux);
                    }
                  ;
dentro_funcao   : var_declaracao dentro_funcao
                    {
                        char aux[1000];
                        strcpy(aux, $1);
                        strcat(aux, $2);
                        $$ = strdup(aux);
                    }
                | comando dentro_funcao
                    {
                        char aux[5000];
                        strcpy(aux, $1);
                        strcat(aux, $2);
                        $$ = strdup(aux);
                    }
                | /*vazio*/     { $$ = strdup(""); }
                  ;
comando             : expressao_decl
                | composto_decl
                | selecao_decl
                | iteracao_decl
                | para_decl
                | facaenq_decl
                | escolha_decl
                | leia_decl
                | escreva_decl
                | retorno_decl
                | interrompa_decl
                  ;
expressao_decl  : expressao PONTOEVIRGULA
                    {
                            char aux[1000];
                            strcpy(aux, $1);
                            strcat(aux, ";\n");
                            $$ = strdup(aux);
                    }
                | PONTOEVIRGULA     { $$ = strdup(";\n"); }
                  ;
selecao_decl    : SE ABREPARENTESE expressao FECHAPARENTESE comando %prec LOWER_THAN_SENAO
                    {
                        char aux[1000];
                        strcpy(aux, "if (");
                        strcat(aux, $3);
                        strcat(aux, ")");
```

```
                        strcat(aux, $5);
                        $$ = strdup(aux);
                    }
                | SE ABREPARENTESE expressao FECHAPARENTESE comando SENAO comando
                    {
                        char aux[1000];
                        strcpy(aux, "if (");
                        strcat(aux, $3);
                        strcat(aux, ")");
                        strcat(aux, $5);
                        strcat(aux, "else ");
                        strcat(aux, $7);
                        $$ = strdup(aux);
                    }
                ;
iteracao_decl   : ENQUANTO ABREPARENTESE expressao FECHAPARENTESE comando
                    {
                        char aux[1500];
                        strcpy(aux, "while (");
                        strcat(aux, $3);
                        strcat(aux, ")");
                        strcat(aux, $5);
                        $$ = strdup(aux);
                    }
                  ;

para_decl       : PARA ABREPARENTESE id ATRIBUICAO expressao PONTOEVIRGULA
                  expressao PONTOEVIRGULA expressao_unaria FECHAPARENTESE comando
                        {
                            char aux[1000];
                            char operacao[20];
                            char passo[20];
                            int temp = atoi($9);
                            if(temp >= 0){
                                strcpy(operacao, $3);
                                strcat(operacao, "<= ");
                                strcat(operacao, $7);
                                strcpy(passo, $3);
                                strcat(passo, "=");
                                strcat(passo, $3);
                                strcat(passo, "+ ");
                                strcat(passo, $9);
                            } else {
                                strcpy(operacao, $3);
                                strcat(operacao, ">= ");
                                strcat(operacao, $7);
                                strcpy(passo, $3);
                                strcat(passo, "=");
                                strcat(passo, $3);
                                strcat(passo, $9);
                            }
                            strcpy(aux, "for (");
                            strcat(aux, $3);
                            strcat(aux, " = ");
                            strcat(aux, $5);
                            strcat(aux, "; ");
                            strcat(aux, operacao);
                            strcat(aux, "; ");
                            strcat(aux, passo);
                            strcat(aux, ") ");
                            strcat(aux, $11);
                            $$ = strdup(aux);
                        }
                    ;
facaenq_decl    : FACA comando ENQUANTO ABREPARENTESE expressao FECHAPARENTESE PONTOEVIRGULA
                    {
                        char aux[1500];
                        strcpy(aux, "do");
                        strcat(aux, $2);
                        strcat(aux, "while (");
                        strcat(aux, $5);
                        strcat(aux, ");\n");
                        $$ = strdup(aux);
                    }
                  ;
escolha_decl    : ESCOLHA ABREPARENTESE var FECHAPARENTESE ABRECHAVE
                  escolhas_dentro padrao_opc FECHACHAVE
                    {
                        char aux[1000];
                        strcpy(aux, "switch (");
                        strcat(aux, $3);
                        strcat(aux, ") {\n");
                        strcat(aux, $6);
                        strcat(aux, $7);
                        strcat(aux, "}\n");
                        int i;
                        for(i = 0; i<current_tab_tipo ; i++){
                            if(strstr($3, tabela[i].nome) != NULL){
                                if(strcmp(tabela[i].tipo, "char ") == 0){
                                    tabela[i].flag_char = 1;
                                    tabela[i].flag_string = 0;
                                }
                            }
                        }
                        $$ = strdup(aux);
                    }
                  ;
leia_decl   : LEIA ABREPARENTESE string VIRGULA var FECHAPARENTESE PONTOEVIRGULA
                {
                    char *result;
                    char aux[400];
                    char aux2[400];
                    if(strstr($3, "%texto") != NULL){
                        strcpy(aux, "fflush(stdin);\ngets(");
                        strcat(aux, $5);
                        strcat(aux, ");\n");
                    } else {
                        result = repl_str($3,"%inteiro","%d");
                        result = repl_str(result,"%real","%f");
                        strcpy(aux, "scanf(");
                        strcat(aux, result);
                        strcat(aux, ", &");
                        strcat(aux, $5);
                        strcat(aux, ");\n");
                    }
                    $$ = strdup(aux);
                    free(result);
                }
                  ;
escreva_decl    : ESCREVA ABREPARENTESE string escreva_fator FECHAPARENTESE PONTOEVIRGULA
                    {
                        char *result;
                        char aux[400];
                        result = repl_str($3,"%inteiro","%d");
                        result = repl_str(result,"%real","%f");
```

```
                        result = repl_str(result,"%texto","%s");
                        result = repl_str(result,"%.5real","%.5f");
                        strcpy(aux, "printf (");
                        strcat(aux, result);
                        strcat(aux, $4);
                        strcat(aux, ");\n");
                        $$ = strdup(aux);
                        free(result);
                    }
                  ;
maiuscula_decl  : MAIUSCULA ABREPARENTESE char_fator FECHAPARENTESE
                    {
                        char aux[200];
                        strcpy(aux, "toupper (");
                        strcat(aux, $3);
                        strcat(aux, ")");
                        $$ = strdup(aux);
                    }
                  ;
concatena_decl  : CONCATENA ABREPARENTESE concatena_fator VIRGULA concatena_fator FECHAPARENTESE
                    {
                        char aux[200];
                        strcpy(aux, "strcat(");
                        strcat(aux, $3);
                        strcat(aux, ", ");
                        strcat(aux, $5);
                        strcat(aux, ")");
                        $$ = strdup(aux);
                    }
                  ;
interrompa_decl     : INTERROMPA PONTOEVIRGULA      { $$ = strdup("break;\n"); }
                  ;
concatena_fator     : string
                | var
                  ;
retorno_decl    : RETORNE PONTOEVIRGULA     { $$ = strdup("return;\n"); }
                | RETORNE expressao PONTOEVIRGULA
                    {
                        char aux[50];
                        strcpy(aux, "return ");
                        strcat(aux, $2);
                        strcat(aux, ";\n");
                        $$ = strdup(aux);
                    }
                  ;
escreva_fator   : VIRGULA expressao escreva_fator
                    {
                        char aux[100];
                        strcpy(aux, ", ");
                        strcat(aux, $2);
                        strcat(aux, $3);
                        $$ = strdup(aux);
                    }
                | /*vazio*/     { $$ = strdup(""); }
                  ;
char_fator          : var
                | caractere
                  ;
escolhas_dentro     : CASO escolha_simples escolhas_dentro_mais
                      DOISPONTOS comando_caso escolhas_dentro
                        {
                            char aux[400];
                            strcpy(aux, "case ");
                            strcat(aux, $2);
                            strcat(aux, $3);
                            strcat(aux, ":\n");
                            strcat(aux, $5);
                            strcat(aux, "break;\n");
                            strcat(aux, $6);
                            $$ = strdup(aux);
                        }
                | /*vazio*/     { $$ = strdup(""); }
                  ;
escolhas_dentro_mais    : VIRGULA escolha_simples escolhas_dentro_mais
                            {
                                char aux[100];
                                strcpy(aux, ":\ncase ");
                                strcat(aux, $2);
                                strcat(aux, $3);
                                $$ = strdup(aux);
                            }
                | /*vazio*/     { $$ = strdup(""); }
                  ;
comando_caso            :  /*vazio*/        { $$ = strdup(""); }
                | comando
                  ;
padrao_opc      : PADRAO DOISPONTOS comando
                    {
                        char aux[400];
                        strcpy(aux, "default :\n");
                        strcat(aux, $3);
                        $$ = strdup(aux);
                    }
                | /*vazio*/     { $$ = strdup(""); }
                  ;
expressao       : var ATRIBUICAO expressao
                {
                   char aux[300];
                   strcpy(aux, $1);
                   strcat(aux, " = ");
                   strcat(aux, $3);
                   int i;
                   for(i = 0; i<current_tab_tipo ; i++){
                       if(strcmp($1, tabela[i].nome) == 0){
                           if((tabela[i].flag_string == 1) && (tabela[i].flag_char != 1)){
                               strcpy(aux, "strcpy(");
                               strcat(aux, $1);
                               strcat(aux, ", ");
                               strcat(aux, $3);
                               strcat(aux, ")");
                           }
                       }
                   }
                   $$ = strdup(aux);
                }
                | expressao_logica
                  ;
var     : id var_fat
            {
                char aux[50];
                strcpy(aux, $1);
                strcat(aux, $2);
                $$ = strdup(aux);
            }
```

```
                ;
var_fat     : ABRECOLCHETE expressao FECHACOLCHETE var_fat
                {
                    char aux[100];
                    strcpy(aux, "[");
                    strcat(aux, $2);
                    strcat(aux, "]");
                    strcat(aux, $4);
                    $$ = strdup(aux);
                }
                | /*vazio*/     { $$ = strdup(""); }
                ;
expressao_logica    : OU termo_logico expressao_logica_fat
                        {
                            char aux[200];
                            strcpy(aux, "||");
                            strcat(aux, $2);
                            strcat(aux, $3);
                            $$ = strdup(aux);
                        }
                | termo_logico expressao_logica_fat
                    {
                        char aux[200];
                        strcpy(aux, $1);
                        strcat(aux, $2);
                        $$ = strdup(aux);
                    }
                ;
expressao_logica_fat    : OU termo_logico expressao_logica_fat
                            {
                                char aux[200];
                                strcpy(aux, "||");
                                strcat(aux, $2);
                                strcat(aux, $3);
                                $$ = strdup(aux);
                            }
                | /*vazio*/     { $$ = strdup(""); }
                ;
termo_logico    : expressao_simples termo_logico_fat
                    {
                        char aux[200];
                        strcpy(aux, $1);
                        int i;
                        for(i = 0; i<current_tab_tipo ; i++){
                            if(strstr($1, tabela[i].nome) != NULL){
                                if(strstr($1, "\'") != NULL){
                                    if(strstr($1, "[") == NULL){
                                        if((strstr($1, "!=") != NULL) || strstr($1, "==") != NULL){
                                            tabela[i].flag_char = 1;
                                            tabela[i].flag_string = 0;
                                        }
                                    }
                                }
                            }
                        }
                        strcat(aux, $2);
                        $$ = strdup(aux);
                    }
                ;
termo_logico_fat    : E expressao_simples termo_logico_fat
                        {
                            char aux[200];
                            strcpy(aux, "&&");
                            strcat(aux, $2);
                            strcat(aux, $3);
                            $$ = strdup(aux);
                        }
                | /*vazio*/     { $$ = strdup(""); }
                ;

expressao_simples   : expressao_soma expressao_simples_fat
                        {
                            char aux[200];
                            strcpy(aux, $1);
                            strcat(aux, $2);
                            int i;
                            for(i = 0; i<current_tab_tipo ; i++){
                                if(strcmp($1, tabela[i].nome) == 0){
                                    if((tabela[i].flag_string == 1) && (tabela[i].flag_char != 1)){
                                        if(strstr($2, "==") != NULL){
                                            if(strstr($2, "\'") == NULL){
                                                char *result;
                                                strcpy(aux, "strcmp(");
                                                strcat(aux, $1);
                                                result = repl_str($2, "==", "");
                                                strcat(aux, ", ");
                                                strcat(aux, result);
                                                strcat(aux, ") == 0");
                                            }
                                        }
                                    }
                                }
                            }
                            $$ = strdup(aux);
                        }
                | EXCLAMACAO expressao_soma expressao_simples_fat
                    {
                        char aux[200];
                        strcpy(aux, "!");
                        strcat(aux, $2);
                        strcat(aux, $3);
                        $$ = strdup(aux);
                    }
                | VERDADEIRO    { $$ = strdup("true"); }
                | FALSO     { $$ = strdup("false"); }
                ;
relacional          : RELACIONAL        { $$ = strdup(yytext); }
                ;
expressao_simples_fat   : relacional expressao_soma
                            {
                                char aux[200];
                                strcpy(aux, $1);
                                strcat(aux, $2);
                                $$ = strdup(aux);
                            }
                | /*vazio*/     { $$ = strdup(""); }
                ;
somasub             : SOMASUB   { $$ = strdup(yytext); }
                ;
expressao_soma      : somasub termo expressao_soma_fat
                        {
                            char aux[200];
                            strcpy(aux, $1);
                            strcat(aux, $2);
```

```
                            strcat(aux, $3);
                            $$ = strdup(aux);
                        }
                | termo expressao_soma_fat
                        {
                            char aux[200];
                            strcpy(aux, $1);
                            strcat(aux, $2);
                            $$ = strdup(aux);
                        }
                ;
expressao_soma_fat  : somasub termo expressao_soma_fat
                        {
                            char aux[200];
                            strcpy(aux, $1);
                            strcat(aux, $2);
                            strcat(aux, $3);
                            $$ = strdup(aux);
                        }
                | /*vazio*/     { $$ = strdup(""); }
                ;
termo   : fator termo_fat
            {
                char aux[200];
                strcpy(aux, $1);
                strcat(aux, $2);
                $$ = strdup(aux);
            }
                ;
multdiv     : MULTDIV       { $$ = strdup(yytext); }
                ;
termo_fat   : multdiv fator termo_fat
                {
                    char aux[200];
                    strcpy(aux, $1);
                    strcat(aux, $2);
                    $$ = strdup(aux);
                }
                | /*vazio*/     { $$ = strdup(""); }
                ;

fator   : ABREPARENTESE expressao FECHAPARENTESE
            {
                char aux[200];
                strcpy(aux, "(");
                strcat(aux, $2);
                strcat(aux, ")");
                $$ = strdup(aux);
            }
                | var
                | ativacao
                | numreal
                | numint
                | caractere
                | string
                | raizq_decl
                | potencia_decl
                | tamanhotexto_decl
                | maiuscula_decl
                | concatena_decl
                ;
expressao_unaria    : numint
                | somasub numint        {
                                            char aux[30];
                                            strcpy(aux, $1);
                                            strcat(aux, $2);
                                            $$ = strdup(aux);
                                        }
                ;
escolha_simples     : expressao_unaria
                | caractere
                ;
raizq_decl  : RAIZQUADRADA ABREPARENTESE expressao_soma FECHAPARENTESE
                {
                    char aux[150];
                    strcpy(aux, "sqrt (");
                    strcat(aux, $3);
                    strcat(aux, ")");
                    $$ = strdup(aux);
                }
                ;
potencia_decl   : POTENCIA ABREPARENTESE expressao_soma VIRGULA expressao_soma FECHAPARENTESE
                    {
                        char aux[100];
                        strcpy(aux, "pow (");
                        strcat(aux, $3);
                        strcat(aux, ", ");
                        strcat(aux, $5);
                        strcat(aux, ")");
                        $$ = strdup(aux);
                    }
                ;
tamanhotexto_decl   : TAMANHOTEXTO ABREPARENTESE texto_fator FECHAPARENTESE
                        {
                            char aux[100];
                            strcpy(aux, "strlen (");
                            strcat(aux, $3);
                            strcat(aux, ")");
                            $$ = strdup(aux);
                        }
                ;
texto_fator         : var
                | string
                ;
ativacao    : id ABREPARENTESE args FECHAPARENTESE
                {
                    char aux[100];
                    strcpy(aux, $1);
                    strcat(aux, "(");
                    int i;
                    for(i = 0; i<current_tab_tipo ; i++){
                        if(strstr($3, tabela[i].nome) != NULL){
                            if(strstr(tabela[i].vet, "[") != NULL){
                                tabela[i].flag_param = 1;
                            }
                        }
                    }
                    strcat(aux, $3);
                    strcat(aux, ")");
                    $$ = strdup(aux);
                }
                ;
args    : arg_lista
```

```
                | /*vazio*/     { $$ = strdup(""); }
                ;
arg_lista   : expressao args_fat
                {
                    char aux[100];
                    strcpy(aux, $1);
                    strcat(aux, $2);
                    $$ = strdup(aux);
                }
              ;
args_fat    : VIRGULA expressao args_fat
                {
                    char aux [100];
                    strcpy(aux, ", ");
                    strcat(aux, $2);
                    strcat(aux, $3);
                    $$ = strdup(aux);
                }
                | /*vazio*/     { $$ = strdup(""); }
                ;
%%
void yyerror (char *s) {
    fprintf(stderr, "%s\n", s);
}

int main(int argc, char *argv[]){
    yyout = fopen(argv[2],"w");
    fprintf(yyout,"#include <stdio.h>\n");
    fprintf(yyout,"#include <stdlib.h>\n");
    fprintf(yyout,"#include <stdbool.h>\n");
    fprintf(yyout,"#include <math.h>\n");
    fprintf(yyout,"#include <string.h>\n");
    fprintf(yyout,"\n");
    yyin = fopen(argv[1], "r");
    yyparse();
    return 0;
}
```