# Escolha do tema para o projeto de Tradutores: Linguagem CStr

Eduardo Furtado — 09/0111575
Departamento de Ciência da Computação
Universidade de Brasília

22 de agosto de 2016

## 1 Área da Ciência da Computação

A área escolhida é algoritmos e estruturas de dados em strings. Propõe-se a implementação de um tradutor da linguagem C (CStr) para código de 3 endereços que implemente o tipo primitivo string com operadores e métodos auxiliares.

## 2 Descrição da linguagem formal para a qual será implementado o tradutor

Para fazer o processo de tradução utilizarei ferramentas como o bison e o flex, bem como o gcc para compilar o tradutor que será desenvolvido em linguagem C.

Este trabalho apresenta uma linguagem CStr, a gramática de CStr é baseada em uma versão reduzida da linguagem C [1], com acréscimo do tipo primitivo string e o operador de concatenação (.), e métodos segundo [2].

A gramática livre-do-contexto a seguir descreve a linguagem CStr proposta, onde variáveis (simbolos não-terminais) começam com letras maiúsculas, Function é a variável inicial e todos os outros simbolos são terminais. A barra vertical | é usada para indicar definições alternativas para um não-terminal.

| | | |
|---|---|---|
| Function | → | Type Identifier ( FormalArgList ) CompoundStmt |
| Identifier | → | char ( char \| digit \| _ )* |
| FormalArgList | → | FormalArg |
| | | \| FormalArgList , FormalArg |
| FormalArg | → | Type Identifier |
| ArgList | → | Arg |
| | | \| ArgList , Arg |
| Arg | → | Identifier |
| Declaration | → | Type IdentList ; |

| | |
|---|---|
| Type → | int |
| | \| float |
| | \| string |
| StringFunction → | string . Identifier () |
| | \| string . Identifier ( ArgList ) |
| StringConcat → | string . string |
| IdentList → | Identifier , IdentList |
| | \| Identifier |
| Stmt → | WhileStmt |
| | \| Expr ; |
| | \| IfStmt |
| | \| CompoundStmt |
| | \| Declaration |
| | \| StringFunction |
| | \| StringConcat |
| | \| ; |
| WhileStmt → | while ( Expr ) Stmt |
| IfStmt → | if ( Expr ) Stmt ElsePart |
| ElsePart → | else Stmt |
| | \| $\varepsilon$ |
| CompoundStmt → | { StmtList } |
| StmtList → | StmtList Stmt |
| | \| $\varepsilon$ |
| Expr → | Identifier = Expr |
| | \| Rvalue |
| Rvalue → | Rvalue Compare Mag |
| | \| Mag |
| Compare → | == |
| | \| < |
| | \| > |
| | \| <= |
| | \| >= |
| | \| ! = |
| Mag → | Mag + Term |
| | \| Mag - Term |
| | \| Term |
| Term → | Term * Factor |
| | \| Term / Factor |
| | \| Factor |
| Factor → | ( Expr ) |
| | \| - Factor |
| | \| + Factor |
| | \| Identifier |
| | \| number |

A ordem em que as operações aparecem determina a precedência de cada operador, onde a primeira tem menor procedência e a última tem a maior procedência.

# 3 Motivação para a escolha da linguagem

Quando se compara C com outras linguagens de programação, como por exemplo C++, fica evidente que uma das facilidades que o C não tem é no tratamento de strings, algo extremamente abrangente na computação.

Entretanto, C é uma linguagem extremamente usada, e sempre que um programador precisa manipular strings, passa por sua cabeça que seria mais fácil estar utilizando outra linguagem.

Este trabalho pode ser útil ao programador da linguagem e ao usuário leigo para facilitar e agilizar a programação em C quando é necessário manipular strings, disponibilizando strings como um novo tipo primitivo e fornecendo operadores e métodos para esse tipo.

# 4 Descrição breve da semântica da linguagem

O tradutor será implementado de acordo com o que for aprendido nas aulas e na bibliografia da matéria, bem como bibliografia oficial do manual do C. [3], [4], [5].

Espera-se implementar:

- String como tipo nativo;
- Operador . para concatenação de strings;
- Funções apresentadas em [2];
- String matching de maneira eficiente segundo [6];

Ou seja, visa-se encorporar no C uma parte de orientação a objetos para o tipo String.

Abaixo está um exemplo de declaração de uma variável string:

```
string nome_da_variavel_string = "Isso é uma variável em C do tipo nativo string!";
```

A partir disso, será possível concatenar strings:

```
string s = "Isso é uma variável em C do tipo nativo string!";
string aux = "!";
nome_da_variavel_string . "!";
nome_da_variavel_string . aux;
```

```
string nome_da_variavel_string = "Isso é uma variável em C do tipo nativo string!";
string aux = "!";
nome_da_variavel_string . "!";
nome_da_variavel_string . aux;
```

Além disso, utilizar métodos, como o de matching proposto em [6]:

```
string s = "Achar uma agulha num palheiro";
string palheiro = s.kmpPreprocess();
string result = palheiro.kmpSearch("agulha");
```

# Referências


[1] MiniC Grammar. http://www2.ufersa.edu.br/portal/view/uploads/setores/184/AppendixA.pdf. [Acessado em 20 de agosto de 2016].

[2] C string lib reference. http://www.cplusplus.com/reference/cstring/. [Acessado em 20 de agosto de 2016].

[3] A.V. AHO, R. Sethi, and S. Lam. *Compiladores: princípios, técnicas e ferramentas*. LONGMAN DO BRASIL, 2008.

[4] The GNU C Reference Manual. https://www.gnu.org/software/gnu-c-manual/gnu-c-manual.html. [Acessado em 20 de agosto de 2016].

[5] Jutta Degener. ANSI C Yacc grammar. http://www.lysator.liu.se/(nobg)/c/ANSI-C-grammar-y.html. [Acessado em 20 de agosto de 2016].

[6] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, 3 edition, 2009.