**BdeX**

Linguagem para Tabelas

Gonçalo Sousa (98152)

João Moura (43889)

Leonardo Freitas (89131)

Pedro Santos (98158)

Rúben Castelhano (97688)

Vasco Santos (98391)

Linguagens Formais e Autómatos

Departamento de Eletrónica, Telecomunicações e Informática

20 de Junho de 2021

Índice

[**Introdução 2**](#_Toc75097359)

[**Linguagem para Leitura e Escrita de Ficheiros 3**](#_Toc75097360)

[**BdeX - Linguagem para Desenvolvimento de Tabelas 4**](#_Toc75097361)

[3.1 Instruções 4](#_Toc75097362)

[3.1.1 Declarações e Atribuições 4](#_Toc75097363)

[3.1.2 Tipos 5](#_Toc75097364)

[3.2 Comentários 5](#_Toc75097365)

[3.3 Palavras Reservadas 6](#_Toc75097366)

[3.4 Operações Aritméticas 6](#_Toc75097367)

[3.5 Instruções 7](#_Toc75097368)

[3.5.1 Create 7](#_Toc75097369)

[3.5.2 Remove 7](#_Toc75097370)

[3.5.3 Join 8](#_Toc75097371)

[3.5.4 Read 8](#_Toc75097372)

[3.5.5 Insert 8](#_Toc75097373)

[3.5.6 Show 8](#_Toc75097374)

[3.5.7 Extract 9](#_Toc75097375)

[3.5.8 Save 9](#_Toc75097376)

[3.5.9 Modify 9](#_Toc75097377)

[**Conclusões 10**](#_Toc75097378)

# Introdução

A utilização de tabelas é cada vez mais frequente para a organização de documentação em formato digital, quer a nível pessoal, quer a nível de grandes empresas. A formatação da informação em tabelas permite facilitar diversas operações, automatizando tarefas diárias que permitem reduzir tempos de trabalho.

Neste contexto, surge ***BdeX***, uma linguagem simples e de rápida aprendizagem, cuja raiz provém de linguagens de mais alto nível. ***BdeX*** destaca-se pelas suas ferramentas específicas para a criação, edição e formatação de tabelas.

Esta linguagem permite declarar variáveis de diferentes tipos, assim como executar operações binárias. Contém ainda instruções para guardar e ler ficheiros de tabelas. A linguagem destino de ***Bdex*** é ***Java***, ou seja, quando um ficheiro de ***BdeX*** é compilado vai dar origem a um ficheiro ***Java***.

No âmbito do desenvolvimento de tabelas, surge a necessidade de guardar e ler ficheiros previamente elaborados. Desta forma, o programador poderá aceder a ficheiros já gravados, para consultar ou editar, ou guardar o trabalho corrente.

O presente relatório apresenta a linguagem ***BdeX*** e a linguagem para leitura e escrita de ficheiros, incluindo exemplos práticos de utilização de ambas as linguagens e um manual de instruções para a sua utilização.

# Linguagem para Leitura e Escrita de Ficheiros

Para que o utilizador possa trabalhar com tabelas em ficheiros, foi criada uma linguagem simples, que tem como objetivo guardar ou ler ficheiros das tabelas desenvolvidas.

Após compilação deste tipo de ficheiros, toda a informação da tabela é escrita para um ficheiro ou lida para uma base de dados que poderá ser posteriormente utilizada na linguagem ***BdeX***.

*//sintaxe de construção de um ficheiro tabela*

table(Coluna >> tipo, Coluna >> tipo)

{

valor,valor;

}

Tabela 2.1: Constituintes de um ficheiro tabela

|  |  |
| --- | --- |
| table | Palavra reservada ‘table’ |
| Coluna | Identificador (pode ser descrito com letras e números) |
| tipo | Apenas pode ser descrito por ‘int’, ‘double’, ‘text’ |
| valor | Pode ser descrito por números inteiros, reais ou Strings |

Desta forma, é fácil compreender que a base de dados será organizada num Mapa de colunas em que cada chave será representada por uma String que identifica a coluna, à qual corresponde um objecto column que por si só é uma Lista de valores.

*//exemplo de uma tabela*

Table (Nmec >> int, Nome >> text, Nota >> double)

{

1223,”Tiago”,12.3;

1334,”Rui”,14.2;

}

Caso surja um erro no programa do utilizador, este será notificado do mesmo.

# BdeX - Linguagem para Desenvolvimento de Tabelas

## 3.1 Instruções

Em ***Bdex*** podem existir instruções de bloco ou instruções de linha. Ambas são terminadas pelo caracter ‘;’.

text a; *## Declaração (Instrução de linha)*

show table; *## Utilização da instrução show (Instrução de linha)*

*## Instrução de bloco*

Table a -> create {

“abc” >> text;

};

### 3.1.1 Declarações e Atribuições

Para se declarar uma variável apenas é necessário especificar o tipo da variável e a sua designação.

O tipo de uma variável não pode ser alterado durante o restante tempo de vida da mesma. Se o utilizador tentar usar uma variável que não foi inicializada irá ser notificado do erro.

text a; *##Variável "a" do tipo text*

Para atribuirmos um valor a uma variável usamos o token ’->’ após indicar qual a variável pretendida e especificamos o seu valor.

text a -> "Hi"; *##Atribuição do valor "Hi" à variável "a"*

int num; *##Criação da variável num*

num -> 10; *##Atribuição do valor 10 à variável num*

### 3.1.2 Tipos

A linguagem ***BdeX*** apresenta os seguintes tipos:

**int**

Representa um número inteiro, equivalente ao int da linguagem destino.

**text**

Forma de representação de texto, delimitada por aspas, equivalente ao tipo String da linguagem destino.

**double**

Representa um número real, equivalente ao double da linguagem destino

**column**

Representa uma coluna de uma tabela, equivalente a uma lista na linguagem destino.

**table**

Representa a tabela, equivalente a uma linkedhashmap da linguagem destino.

## 3.2 Comentários

Como se pode observar a partir do ponto 3.1., na linguagem ***BdeX***apenas são permitidos comentários de linha que são inicializados pelo caractere’##’.

*##Comentário de linha*

## 3.3 Palavras Reservadas

***Bdex***possui 19 palavras reservadas (Tabela 3.1). Nenhuma dessas palavras pode ser utilizado para nome de variável.

Tabela 3.1: Palavras Reservadas

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| int | double | text | column | Table |
| create | insert | remove | extract | modify |
| save | read | show | join | where |
| from | to | in | into |  |

## 3.4 Operações Aritméticas

Todas as operações aritméticas permitidas, e os respetivos tipos que as suportam estão sintetizados na Tabela 3.2.

Tabela 3.2: Operações Aritméticas, suas funções e tipos permitidos

|  |  |  |
| --- | --- | --- |
| Operação | Função | Tipo |
| + | Adição ou Sinal Positivo | int,double |
| – | Subtração ou Sinal Negativo | int,double |
| \ | Divisão | int,double |
| % | Resto da Divisão inteira | int,double |
| \* | Multiplicação | int,double |
| == | Igualdade | int, double, text |
| != | Diferença | int, double, text |
| < | Menor | int, double, text |
| > | Maior | int, double, text |
| <= | Menor ou Igual | int, double, text |
| >= | Maior ou Igual | int, double, text |

## 3.5 Instruções

### 3.5.1 Create

A instrução createé usada para criar tabelas, somente para identificadores do tipo Table.

Delimitado por ‘{};’, esta instrução necessita que o utilizador coloque uma expressão e o seu respetivo tipo, que irão representar cada coluna da tabela.

Exemplo A:

Table tabela -> create {

“item1” >> int;

“item2” >> text;

“item3” >> double;

};

Como se pode ver, dentro do bloco, temos cada nome da coluna, que pode ser text ou identificador, o seu respetivo tipo, apontado por ‘>>’.

Se o utilizador quiser criar apenas uma coluna, pode usar a mesma instrução desde que coloque ‘column’ a seguir.

Exemplo B:

colunaExemplo -> create column >> int;

Em ambos precisamos de atribuir uma variável a instrução e para isso basta usar o token ‘->’. Varias instruções irão precisar do mesmo token.

### 3.5.2 Remove

Esta instrução é usada para remover colunas duma tabela ou eliminar células dentro da condição dada:

remove colToRemove from tableExample; ## Remove coluna da tabela

remove from tableExample where ( col == “conteúdo”); ## Remove células da coluna, uma ou mais condições

### 3.5.3 Join

A instrução ‘join’ devolve a juncão de duas tabelas numa só tabela, sem repetição de colunas, caso alguma coluna tenha nome igual em ambas as tabelas.

tabelaConjunta -> join tabela1 to tabela2;

### 3.5.4 Read

Para ler um ficheiro com uma tabela, o utilizador tem a sua disposição uma instrucao chamada ‘read’. Ela devolve a tabela do ficheiro para uma variável.

Table exemplo -> read from “tabela”;

O programa recebe a string do nome do ficheiro e verifica se existe um ficheiro do tipo ‘.table’ e devolve a tabela resultante.

Também com esta instrução, o programador pode pedir ao utilizador que insira dados:

int a -> read int;

### 3.5.5 Insert

Para poder inserir conteúdo numa tabela, coluna ou coluna numa tabela, foi criado a instrução ‘insert’. O utilizador pode inserir de várias maneiras:

* insert into tableExample in colExample >> “conteúdo”; ## Insere um elemento na coluna da tabela
* insert into tableExample >> “conteúdo”|1|1.2; ## Insere um ou mais elementos
* insert into tableExample >> col1: ”conteúdo”| col2: 2; ## Insere elementos nas colunas especificadas

O utilizador também pode inserir conteúdo numa coluna ou inserir uma coluna numa tabela, nomeando a coluna com o nome que desejar:

* insert colEx >> “newCol” into tableExample; ## Insere coluna numa tabela com novo nome
* insert “conteúdo” into “newCol”; ## Insere elemento na coluna

### 3.5.6 Show

Uma instrução simples que tem como único propósito imprimir para o ecrã o que lhe for pedido, tabela, coluna, inteiro, etc...

show tabela;

### 3.5.7 Extract

Esta instrução tem como propósito extrair colunas ou elementos de uma coluna e devolve uma tabela ou uma coluna. Para fazer uma seleção de colunas ou elementos, damos ao utilizador uma opção para o fazer.

Exemplo:

Table tabela -> extract from tabelaExemplo >> “coluna1” | coluna2;

column coluna -> extract from tabelaExemplo >> “colunaInt” where t < 10;

No primeiro exemplo, retira-se a coluna 1 e a coluna 2 de uma tabela e guarda-se numa tabela.

No segundo, já se coloca uma condição para remover de uma coluna de inteiros valores menores que dez e guarda-se o resultado numa coluna.

Para usar a seleção damos ao utilizador vários comandos condicionais (‘>’,’<’,’<=’,’>=’,’==’).

### 3.5.8 Save

No final de desenvolvimento de tabelas o utilizador pode guardar a tabela num ficheiro, com a instrução ‘save’.

Para isso basta:

save tabela to “exemplo”;

Será criado um ficheiro exemplo.table com a tabela guardada, que posteriormente poderá ser lido com a instrução ‘read’.

### 3.5.9 Modify

A instrução ‘modify’ serve para alterar valores da tabela/coluna de maneira simples.

modify colToModify:value in tableExample where col == value ##modifica os valores das colunas especificadas, uma ou mais condições.

modify colToModify:position in tableExample >> value ##modifica o valor na posição position da coluna especificada por value.

modify position in column >> value ##modifica o valor na posição position de uma coluna

# Conclusões

Através dos conhecimentos adquiridos ao longo do semestre nas aulas teóricas e praticas e a disponibilidade dos professores foi-nos possível ter o suporte necessário para a realização deste projeto da cadeira de LFA.

No geral nem todas as funcionalidades pedidas foram alcançadas, mas os objetivos principais que nos foram propostos foram implementados.

E o mais importante ao longo do desenvolvimento deste projeto adquirimos conhecimentos fundamentais para a realização desta unidade curricular .