UNEMAT - Campus Cáceres

**Trabalho de Algoritmo II**

Guilherme Felipe Pereira

O trabalho a seguir tem como objetivo apresentar um estudo de caso solicitado pelo professor Rodrigo Costa Camargos aos alunos de Algoritmo II no curso de Ciência da computação.

Link do Github: <https://github.com/GuiFeelipeDev/Cacapalavras>

Código:

#include <stdio.h>

#include <string.h>

static int n = 10;

static int n2 = 0;

static char letras[10][10];

static char letras2[10][10];

static int posXvet[10];

static int posYvet[10];

//Tarefa de análise e procura da palavra no input

void procurando(char mat[][10], char palavra[], int k)

{

int tam\_str = strlen(palavra);//Salvo o tamanho da string

int posX, posY, n1 = 0, count = 0;

//Esquerda para direita

//"For" para percorrer as linhas da matriz

for(int i = 0; i < n; i++)

{

/\*"For" para percorrer as colunas da matriz,

usei um calculo simples para economizar memoria,

eu comparo só o necessario, até onde é possivel

existir uma palavra\*/

for(int c = 0; c < (n - tam\_str + 1); c++)

{

/\*Condição 1: Se uma letra "n" do caça palavras

for igual a primeira letra da minha palavra\*/

if(mat[i][c] == palavra[0])

{

//Se a condição for satisfeita...

//Salvo a posição da primeira letra

posX = c;//Coluna

posY = i;//Linha

/\*"For" para a comparação da palavra na matriz,

Percorro da posição onde foi encontrada a primeira letra,

no caso "c", até c + o tamanho da minha string.\*/

for(int x = c; x < (c + tam\_str); x++)

{

//Faço a comparação letraa por letra direto na matriz

if(mat[i][x] == palavra[n1])

{

count++;

n1++;

/\*Ps: professor, não modulei essa parte pois eu

uso diretamente a matriz, eu não salvo como uma palavra.\*/

}

//Se a comparação falhar...

else

{

//Eu encerro a repetição do for para economizar memoria

x = (c + tam\_str);

n1 = 0;

count = 0;

}

}

/\*Ao final, se o tamanho do contador for igual ao tamanho

total da string...\*/

if(count == tam\_str)

{

/\*Eu salvo a posição nos meus vetores de print,

Pos + 1 para facilitar a leitura ao final\*/

posXvet[n2] = posX + 1;

posYvet[n2] = posY + 1;

n2++;

count = 0;

}

}

}

}

system("cls");

//Inicio da bateria de prints de resultado...

printf("String \"%s\":\n", palavra);//a palavra solicitada

//Printo o caça palavras com o ID correspondente

if(k == 0)

{

print\_cacaPalavra(0);

}

else

{

print\_cacaPalavra(1);

}

printf("========================\n");

printf("Esquerda para a direita:\n");

//Se não houver nem uma ocorrencia...

if(n2 == 0)

{

printf("Nem uma ocorrencia!\n");

}

//"For" para printar as ocorrencias

for(int i = 0; i < n2; i++)

{

//prints básicos

printf("Linha = %d\n", posYvet[i]);

printf("Coluna = %d\n", posXvet[i]);

if(i < n2 - 1)//Estética

{

printf("\n");

}

}

printf("========================\n");

//"For" para zerar meus vetores de print

for(int i = 0; i < n2; i++)

{

posXvet[n2] = 0;

posYvet[n2] = 0;

}

//Zero minhas variaveis auxiliares para não ter probelmas

n2 = 0;

n1 = 0;

//=======================

//Direita para a esquerda

//A única mudança nessa parte é na hora de percorrer as colunas...

for(int i = 0; i < n; i++)

{

//Começo do fim da linha, indo até o começo...

for(int c = 9; c > (tam\_str - 2); c--)

{

//A lógica nesssa parte segue a mesma

if(mat[i][c] == palavra[0])

{

posX = c;

posY = i;

for(int x = c; x > (c - tam\_str); x--)

{

if(mat[i][x] == palavra[n1])

{

count++;

n1++;

}

else

{

x = (c - tam\_str);

n1 = 0;

count = 0;

}

}

if(count == tam\_str)

{

posXvet[n2] = posX + 1;

posYvet[n2] = posY + 1;

n2++;

count = 0;

}

}

}

}

printf("========================\n");

printf("Direita para a esquerda:\n");

if(n2 == 0)

{

printf("Nem uma ocorrencia!\n");

}

for(int i = 0; i < n2; i++)

{

printf("Linha = %d\n", posYvet[i]);

printf("Coluna = %d\n", posXvet[i]);

if(i < n2 - 1)

{

printf("\n");

}

}

printf("========================\n");

for(int i = 0; i < n2; i++)

{

posXvet[n2] = 0;

posYvet[n2] = 0;

}

n2 = 0;

n1 = 0;

//=========================

//De cima para baixo

//A unica mudança nessa parte, é a inversao da linha e da coluna

//Esse "For" passa a ser para percorrer as colunas

for(int i = 0; i < n; i++)

{

//Esse "for" passa a percorrer as linhas

for(int c = 0; c < (n - tam\_str + 1); c++)

{

if(mat[c][i] == palavra[0])

{

//Salvo normalmente...

posX = i;

posY = c;

for(int x = c; x < (c + tam\_str); x++)

{

if(mat[x][i] == palavra[n1])

{

count++;

n1++;

}

else

{

x = (c + tam\_str);

n1 = 0;

count = 0;

}

}

if(count == tam\_str)

{

posXvet[n2] = posX + 1;

posYvet[n2] = posY + 1;

n2++;

count = 0;

}

}

}

}

printf("========================\n");

printf("De cima para baixo:\n");

if(n2 == 0)

{

printf("Nem uma ocorrencia!\n");

}

for(int i = 0; i < n2; i++)

{

printf("Linha = %d\n", posYvet[i]);

printf("Coluna = %d\n", posXvet[i]);

if(i < n2 - 1)

{

printf("\n");

}

}

printf("========================\n");

for(int i = 0; i < n2; i++)

{

posXvet[n2] = 0;

posYvet[n2] = 0;

}

n2 = 0;

n1 = 0;

//=======================

//De baixo para cima

//A única coisa que muda nessa parte é na hora de percorrer a coluna

for(int i = 0; i < n; i++)

{

//Começo do final da coluna até o inicio

for(int c = 9; c > (tam\_str - 2); c--)

{

if(mat[c][i] == palavra[0])

{

posX = i;

posY = c;

for(int x = c; x > (c - tam\_str); x--)

{

if(mat[x][i] == palavra[n1])

{

count++;

n1++;

}

else

{

x = (c - tam\_str);

n1 = 0;

count = 0;

}

}

if(count == tam\_str)

{

posXvet[n2] = posX + 1;

posYvet[n2] = posY + 1;

n2++;

count = 0;

}

}

}

}

printf("========================\n");

printf("De baixo para cima:\n");

if(n2 == 0)

{

printf("Nem uma ocorrencia!\n");

}

for(int i = 0; i < n2; i++)

{

printf("Linha = %d\n", posYvet[i]);

printf("Coluna = %d\n", posXvet[i]);

if(i < n2 - 1)

{

printf("\n");

}

}

printf("========================\n");

//=======================

system("pause");

}

//===============================================

//Procedimento básico para printar o caça palavras

void print\_cacaPalavra(int x)

{

//Printo o caça palavras com o ID equivalente

for(int i = 0; i < n; i++)

{

for(int c = 0; c < n; c++)

{

if(x == 0)

{

printf("%c ", letras[i][c]);

}

else

{

printf("%c ", letras2[i][c]);

}

}

printf("\n");

}

}

//=================================================

//Menu básico para coleta de dados

void sub\_menu(int x)

{

system("cls");

char palavra[] = "";

//print do caça palavas segundo o ID

printf("===CACA PALAVRAS===\n");

if(x == 0)

{

print\_cacaPalavra(0);

}

else

{

print\_cacaPalavra(1);

}

//Coleta da palavra a ser pesquisada

printf("Insira a palavra que deseja procurar: \n");

printf(">> ");

scanf("%s", &palavra);

//Chamo a função de procura passando o caça palavras equivalente ao ID

if(x == 0)

{

procurando(letras, palavra, x);

}

else

{

procurando(letras2, palavra, x);

}

}

//================================

//Menu para inserir por meio do teclado

void preencher()

{

system("cls");

//Caça palavras inserido pelo teclado

for(int i = 0; i < n; i++)

{

for(int c = 0; c < n; c++)

{

system("cls");

print\_cacaPalavra(1);

printf("Insira a letra: ");

scanf("%s", &letras2[i][c]);

}

}

//Submenu chamado passando o ID 1

sub\_menu(1);

}

//=====================================

//Seleção de opção do usuário

void menu()

{

/\*Sistema de ID incrementado:

ID 0 = Caça palavras padrão

ID 1 = Caça palavras Inserido pelo teclado

Dependendo da função eu chamo as funções passando

o caça palavras necessário\*/

int selected\_opt;

printf("====================MENU==================\n");

printf("1 - Usar caca-palavras padrao\n");

printf("2 - Inserir um caca-palavras pelo teclado\n");

printf("3 - Sair\n");

printf("==========================================\n");

printf("Insira a opcao: ");

scanf("%d", &selected\_opt);

switch(selected\_opt)

{

case 1:

//Submenu chamado passando o ID 0

sub\_menu(0);

break;

case 2:

preencher();

break;

case 3:

system("cls");

printf("Obrigado por usar o meu programa!\n");

system("pause");

break;

default:

printf("Selecione uma opcao valida!\n");

system("pause");

menu();

break;

}

}

//===========================

int main()

{

//Estrutura do caça palavras

letras[0][0]='a'; letras[0][1]='b'; letras[0][2]='a'; letras[0][3]='c'; letras[0][4]='a'; letras[0][5]='x'; letras[0][6]='a'; letras[0][7]='b'; letras[0][8]='a'; letras[0][9]='a';

letras[1][0]='c'; letras[1][1]='b'; letras[1][2]='y'; letras[1][3]='f'; letras[1][4]='a'; letras[1][5]='z'; letras[1][6]='a'; letras[1][7]='a'; letras[1][8]='a'; letras[1][9]='l';

letras[2][0]='i'; letras[2][1]='t'; letras[2][2]='a'; letras[2][3]='e'; letras[2][4]='h'; letras[2][5]='p'; letras[2][6]='e'; letras[2][7]='r'; letras[2][8]='o'; letras[2][9]='a';

letras[3][0]='p'; letras[3][1]='i'; letras[3][2]='r'; letras[3][3]='a'; letras[3][4]='v'; letras[3][5]='e'; letras[3][6]='r'; letras[3][7]='o'; letras[3][8]='f'; letras[3][9]='f';

letras[4][0]='o'; letras[4][1]='z'; letras[4][2]='t'; letras[4][3]='a'; letras[4][4]='z'; letras[4][5]='m'; letras[4][6]='a'; letras[4][7]='d'; letras[4][8]='i'; letras[4][9]='f';

letras[5][0]='i'; letras[5][1]='x'; letras[5][2]='q'; letras[5][3]='m'; letras[5][4]='u'; letras[5][5]='a'; letras[5][6]='l'; letras[5][7]='a'; letras[5][8]='l'; letras[5][9]='i';

letras[6][0]='t'; letras[6][1]='c'; letras[6][2]='w'; letras[6][3]='n'; letras[6][4]='u'; letras[6][5]='s'; letras[6][6]='u'; letras[6][7]='l'; letras[6][8]='g'; letras[6][9]='n';

letras[7][0]='t'; letras[7][1]='e'; letras[7][2]='s'; letras[7][3]='t'; letras[7][4]='e'; letras[7][5]='d'; letras[7][6]='e'; letras[7][7]='l'; letras[7][8]='i'; letras[7][9]='n';

letras[8][0]='h'; letras[8][1]='o'; letras[8][2]='m'; letras[8][3]='a'; letras[8][4]='r'; letras[8][5]='o'; letras[8][6]='b'; letras[8][7]='o'; letras[8][8]='b'; letras[8][9]='a';

letras[9][0]='k'; letras[9][1]='o'; letras[9][2]='a'; letras[9][3]='z'; letras[9][4]='z'; letras[9][5]='z'; letras[9][6]='a'; letras[9][7]='m'; letras[9][8]='a'; letras[9][9]='l';

menu();

return 0;

}