**Universidade Tecnológica Federal do Paraná**

**Departamento acadêmico de eletrônica (DAELN)**

Curso: Engenharia Mecatrônica

Disciplina: Fundamentos de Programação – EL72E / Turmas S15 e S16

Professores: Edenilson José da Silva, Omero Francisco Bertol e Valfredo Pilla Júnior

Data: \_\_/\_\_/\_\_\_\_\_

**Prova 1**

NOTA

**Aluno:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Turma:\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

1. **(Valor: 3,5)** (**funções, arrays**). Supor que uma “imagem” é composta por valores numéricos 0 e 1 em uma matriz de 4 linhas e 4 colunas. Elabora uma função que localiza “cantos” nestas imagens. A Figura 1 (a), (b) e (c) apresentam exemplos de “imagens” e a Figura 2 a única forma que caracteriza um “canto”.

Figura 1 – Imagens com “cantos”.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | |  | 0 | 1 | 2 | 3 | | 0 | 0 | 0 | 1 | 1 | | 1 | 1 | 1 | 1 | 0 | | 2 | 1 | 0 | 1 | 0 | | 3 | 0 | 0 | 0 | 0 |  1. Imagem com 2 “cantos” | |  |  |  |  |  | | --- | --- | --- | --- | --- | |  | 0 | 1 | 2 | 3 | | 0 | 1 | 1 | 0 | 0 | | 1 | 1 | 0 | 0 | 0 | | 2 | 0 | 0 | 1 | 1 | | 3 | 0 | 0 | 1 | 0 |  1. Imagem com 2 “cantos” | |  |  |  |  |  | | --- | --- | --- | --- | --- | |  | 0 | 1 | 2 | 3 | | 0 | 1 | 1 | 0 | 0 | | 1 | 1 | 1 | 1 | 0 | | 2 | 0 | 0 | 1 | 1 | | 3 | 0 | 0 | 1 | 0 |  1. Imagem com 1 “canto” |

Figura 2 – Forma do “canto”.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | |  |  | | --- | --- | | 1 | 1 | | 1 | 0 | | Em um canto as seguintes relações são verdadeiras:  imagem[i][j] == 1  imagem[i][j+1] == 1  imagem[i+1][j] == 1  imagem[i+1][j+1] == 0 |

Neste contexto, elabore uma função **int conta\_cantos(int imagem[4][4])**, onde imagem é a matriz 4x4 que contém a imagem, e retorna um inteiro correspondente ao número de “cantos” localizados na imagem. Elaborar um programa que solicita ao usuário a matriz que corresponde à imagem e retorna o número de cantos localizados.

Implementar na função “main”:

a) entrada dos dados da imagem (armazenar na matriz)

b) mostrar a matriz com os dados da imagem

c) chamar e mostrar a quantidades de cantos calculado pela função “conta\_cantos”

1. **(Valor: 3,5)** (**funções, arrays, strings**). Desenvolver um corretor ortográfico simples. O corretor ortográfico apresenta a forma correta da palavra se a mesma estiver disponível em seu dicionário interno. Como funciona o corretor: solicita-se ao usuário uma palavra, restrita a 10 caracteres. O programa verifica quantos caracteres da palavra digitada estão presentes em todas as palavras do dicionário. A palavra que possui a maior porcentagem de coincidência é escolha como a “versão correta da palavra”. A resposta deve indicar a porcentagem de coincidência e a palavra selecionada no dicionário.

Exemplos:

Dicionário: bicicletas, amarelados, adocicadas, computador.

>> Qual a palavra? 1º caso de teste

>> **computador**

>> 100% - A palavra correta eh computador

>> Qual a palavra? 2º caso de teste

>> **c**0**mput**3**d**i**r**

>> 70% - A palavra correta eh computador

>> Qual a palavra? 3º caso de teste

>> **comp**rei

>> 40% - A palavra correta eh: computador

Obs: nos exemplos estão em **negrito** os caracteres considerados como pertencentes à palavra existente no dicionário.

Para tanto, elabore uma função **int acha\_palavra(char string[11], char palavra\_dicionario[11])** que devolve a porcentagem (como inteiro de 0 a 100) de correspondência entra a string fornecida pelo usuário e a palavra do dicionário: se (string[i] == palavra\_dicionario[i] então é uma correspondência. Ainda, implemente o programa que faça uso da função e opere como o corretor ortográfico simples. O dicionário faz parte do código deste programa.

Dica - Analise o seguinte código:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

int main()

{

char dicionario[4][11];

strcpy(dicionario[0], "bicicletas");

strcpy(dicionario[1], "amarelados");

strcpy(dicionario[2], "adocicadas");

strcpy(dicionario[3], "computador");

printf("%s", dicionario[1]); // 2ª palavra do dicionário (amarelados)

printf("\n%c", dicionario[0][1]); // 2º caractere da 1ª palavra

return 0;

}

Implementar na função “main”:

a) carregar as palavras do dicionário (strcpy)

b) entrar com a palavra a ser comparada com as palavras do dicionários

c) chamar a função “acha\_palavra” para cada palavra do dicionário. Manter o maio valor de correspondência de caracteres e a posição da palavra com esta correspondência no dicionário.

d) mostrar o % de correspôncia e qual a palavra correta.

1. **(Valor: 3,5)** (**struct, funções, arrays**). Desenvolver um programa para verificar se um número inteiro, chamado de número da sorte, existe em um conjunto de valores e quantos valores deste conjunto são inferiores que o referido número da sorte.

Estrutura de dados sugerida:

typedef struct {

int existe; **// -1: não existe; 0..(n-1) posição do número da sorte no conjunto (não**

**// haverá mais de um número da sorte)**

int contaMenores;

} rgDados;

Funcão “main”:

1. Declarar o conjunto de valores: #define n 9

int conjunto[n] = {**5**, 7, 8, **0**, **2**, 9, **6**, **1**, 15};

1. Solicitar ao usuário o número da sorte, por exemplo: int nroSorte;

scanf(“%d”, &nroSorte);

1. Chamar a função “verificar” e exibir os resultados como sugerido a seguir (supor que na entrada de dados o número da sorte fornecido pelo usuário foi “7”);

**Conjunto: 0:5; 1:7; 2:8; 3:0; 4:2; 5:9; 6:6; 7:1; 8:15;**

**Número da Sorte: 7 (existe na posicao 1).**

**Existem 5 valores infeiros ao número da sorte (55.56% do total).**

Função “verificar”:

1. Interface: rgDados verificar(int x[], int tam, int nroSorte) {...}

Onde: x: conjunto de valores

tam: tamanho do conjunto

nroSorte: número da sorte

1. Objetivo: verifica se o número da sorte existe no conjunto “x” e quantos valores do conjunto são inferiores ao referido número
2. Retornar os resultados através de uma variável registro “rgDados”