http://www.nickgentry.com/

Algoritmos e Programação de Computadores

Disciplina 113476

Prof. Alexandre Zaghetto
http://alexandre.zaghetto.com
zaghetto@unb.br

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Prática de Laboratório 02.a
Vetores

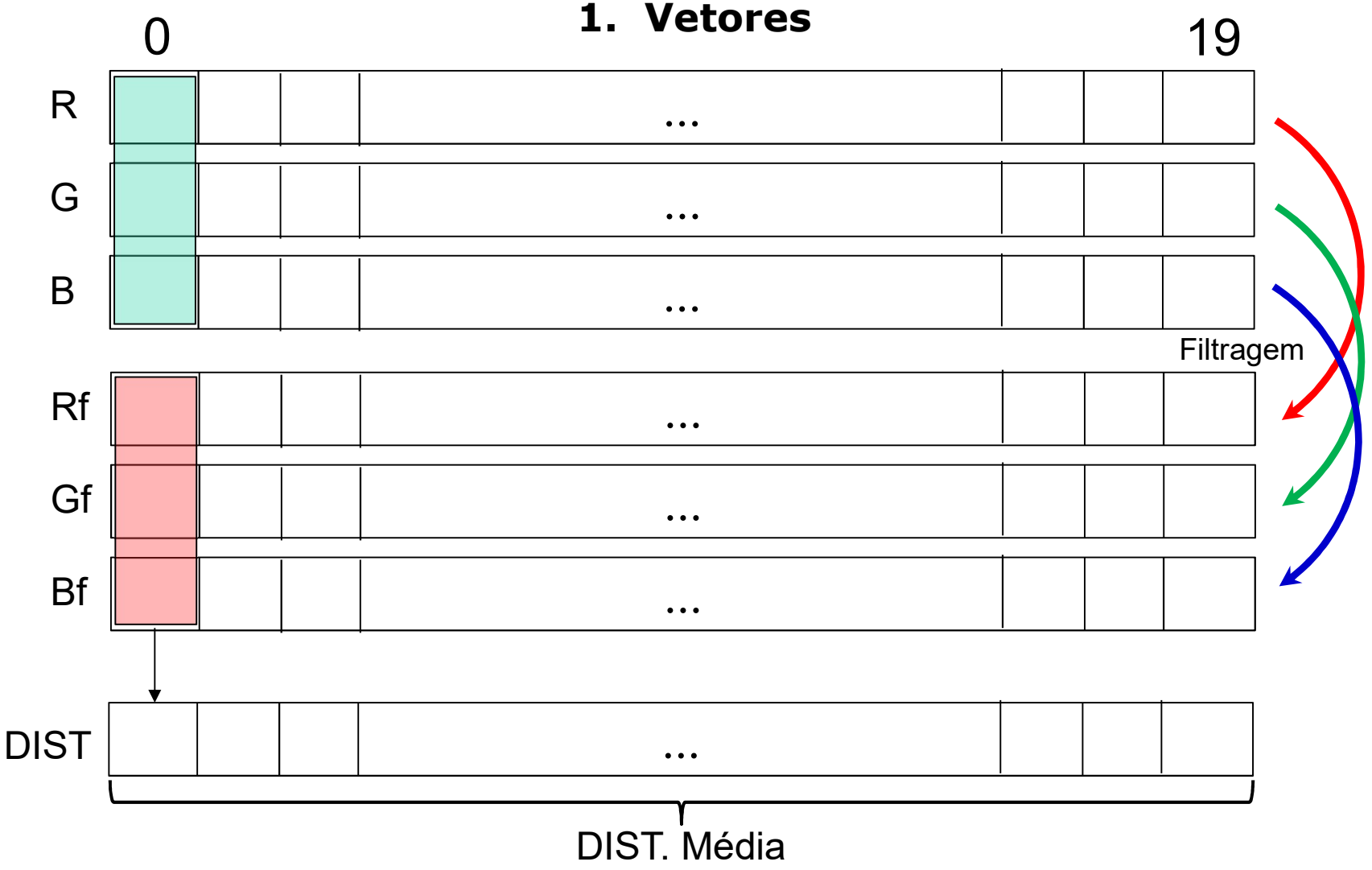
1. Vetores

Problema 1a: Escreva uma programa em linguagem C que solicita ao usuário 20 componentes RGB do tipo int entre 0 e 255 que são armazenadas em três vetores R, G e B. Em seguida, os valores de cada vetor são filtrados por meio da filtragem de média móvel central [1] e o resultado é armazenado em três novos vetores Rf, Gf, Bf. O tamanho da janela de filtragem é fixo e igual a 3. O programa deve calcular também a distância euclidiana [2] média entre os dois vetores RGB e RfGfBf.

Problema 1b: Escreva a solução para o problema 1a utilizando a linguagem Python.

[1] https://en.wikipedia.org/wiki/Moving_average
[2] https://en.wikipedia.org/wiki/Euclidean_distance

1. Vetores



Prática de Laboratório 02.b
Matrizes

2. Matrizes

Problema 2: *The Game of Life*, invented by John Conway in 1970, is an example of a zero-player "game" known as a cellular automaton. The game consists of a two-dimensional world extending infinitely in all directions, divided into "cells." Each cell is either "dead" or "alive" at a given "generation." The game consists of a set of rules that describe how the cells evolve from generation to generation. These rules calculate the state of a cell in the next generation as a function of the states of its neighboring cells in the current generation. In a 2-D world, a cell's neighbors are those 8 cells vertically, horizontally, or diagonally adjacent to that cell. At each step in time, the following transitions occur:

2. Matrizes

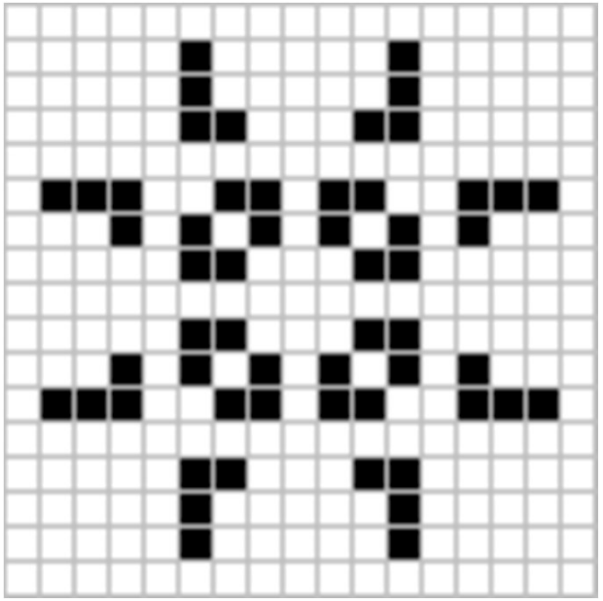
I. Any live cell with fewer than two live neighbors dies, as if caused by under-population.
II. Any live cell with two or three live neighbors lives on the next generation.
III. Any live cell with more than three live neighbors dies, as if by overcrowding.
IV. Any dead cell with exactly three live neighbors becomes a live cell, as if by reproduction.

In this lab, we will implement Conway's Game of Life, with the minor restriction that our 2-D world is finite. The neighbors of a cell on the edge of the world that would be beyond the edge are assumed dead. You can read more about Conway's Game of Life on Wikipedia at http://en.wikipedia.org/wiki/Conway%27s_Game_of_Life.

Este problema foi adaptado do curso *6.087: Practical Programming in C* promovido pelo Massachusetts Institute of Technology (MIT) , Department of Electrical Engineering and Computer Science.

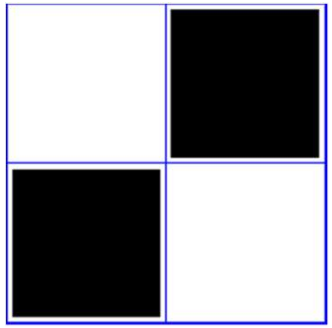
2. Matrizes

Declare uma matriz, definida conforme a ilustração abaixo, que representa a geração inicial do autômato.



2. Matrizes

As células brancas (mortas ou 1) devem ser representadas na tela do computador com quadrados brancos. As células pretas (vivas ou 0) devem ser representadas com quadrados pretos. A grade que delimita as células e o tabuleiro também ser desenhadas utilizando-se linhas azuis. Tanto os quadrados como as linhas devem ser desenhados utilizando-se a biblioteca gráfica playAPC. Veja um exemplo abaixo.



2. Matrizes

Problema 2a: Escreva um programa apenas em linguagem C que implementa o jogo da vida e mostra o estado de cada geração utilizando as funções da PlayCB (http://playapc.zaghetto.com/).

Problema 2b: Escreva a solução para o problema 2a utilizando a linguagem Python.

Prática de Laboratório 02.c
Strings & Structs

3. Strings & Structs

Problema 3: Escreva um programa em linguagem C para criar um cadastro de vários alunos, incluindo a media como campo do cadastro. Para isto, defina um novo tipo de dado (struct) taluno, com os seguintes campos:

- int nr
- char nome[MAX1] (usando define, MAX1 = 100)
- char departamento[MAX2] (usando define, MAX2 = 5)
- float nota[3]
- float media

Declare um vetor aluno[MAX] do tipo taluno, com define MAX = 3.

3. Strings & Structs

Percorra registro a registro (utilizando repetições: for, while ou do...while) para inserir dados em cada variável de cada registro, via teclado, mostrando na tela a mensagem “Digite nr do aluno:”, para que usuário entre com um numero, “Digite nome do aluno:”, para que usuário entre com um nome, “Digite o departamento:”, para que usuário entre com a abreviatura do departamento (CIC ou ENE), “Digite nota 0:”, para que entrar com a nota 0, “Digite nota 1:”, para que entrar com a nota 1 e “Digite nota 2:”, para que entrar com a nota 2.

3. Strings & Structs

A sigla do departamento deve necessariamente ser armazenada em caixa alta (utilizar a função toupper). A primeira letra de cada nome deve ser maiúscula (toupper) e as demais minúsculas (tolower), mesmo que o usuário não respeite esta formatação. Ou seja, tanto no fornecimento da sigla do departamento como no fornecimento do nome, o usuário pode entrar com as informações sem respeitar qualquer critério (maiúscula ou minúscula), cabendo ao programa corrigir essa entrada, de acordo com o critério exposto anteriormente.

Percorrer cada registro do vetor de registros e calcule a média do aluno, a partir dos valores fornecidos para as notas 0, nota 1 e nota 2. Utilizar repetições (for, while ou do...while). O aluno não deve fazer media = (nota[0] + nota[1] + nota[2])/3.

3. Strings & Structs

Imprimir na tela os registros criados:

```
aluno[0].nr : 150
aluno[0].nome : Ana
aluno[0].curso : ENE
aluno[0].nota[0]: 10.0
aluno[0].nota[1]: 8.0
aluno[0].nota[2]: 9.0
aluno[0].media : 9.0

aluno[1].nr : 151
aluno[1].nome : Beto
aluno[1].curso : CIC
aluno[1].nota[0]: 8.0
aluno[1].nota[1]: 10.0
aluno[1].nota[2]: 9.0
aluno[1].media : 9.0

aluno[2].nr : 152
aluno[2].nome : Carla
aluno[2].curso : ENE
aluno[2].nota[0]: 7.0
aluno[2].nota[1]: 10.0
aluno[2].nota[2]: 10.0
aluno[2].media : 9.0
```

3. Strings & Structs

Ao final, criar mecanismos para responder o seguinte:

- A média da turma.
- A menor nota na primeira prova.
- O nome do aluno com maior média.
- O numero de alunos abaixo da media da turma.
- O(s) aluno(s) reprovado(s) foi(ram): imprimir o(s) nome(s) do(s) aluno(s). Considere: reprovação ➔ média < 5.0. Caso não haja reprovação, imprimir “Não há!”.