

Disciplina**: Estrutura de Dados I**

Profª**: Hellen Carmo de Oliveira Matos**

**Lista de Exercícios – Listas Estáticas**

1. Dada uma lista sequencial ordenada L1, escreva funções que:
2. verifique se L1 está ordenada ou não (a ordem pode ser crescente ou decrescente)
3. faça uma cópia da lista L1 em uma outra lista L2;
4. faça uma cópia da Lista L1 em L2, eliminando elementos repetidos;
5. inverta L1 colocando o resultado em L2;
6. inverta L1 colocando o resultado na própria L1;
7. intercale L1 com a lista L2, gerando a lista L3. considere que L1, L2 e L3 são ordenadas.
8. gere uma lista L2 onde cada registro contém dois campos de informação: *elem* contém um elemento de L1, e *count* contém quantas vezes este elemento apareceu em L1.
9. elimine de L1 todas as ocorrências de um elemento dado, L1 ordenada.
10. assumindo que os elementos da lista L1 são inteiros positivos, forneça os elementos que aparecem o maior e o menor número de vezes (forneça os elementos e o número de vezes correspondente)
11. O número de CPF possui nove dígitos mais dois dígitos de controle (DCs). Esses dois dígitos, chamados DC1 e DC2, são calculados da seguinte maneira (para responder essa questão, estudar o material “Vetores e endereços”:

* DC1: Multiplica-se cada um dos nove dígitos do CPF, da direita para a esquerda, por um peso que começa em 2 e é incrementado de 1 a cada dígito, somando-se todos os produtos obtidos. Esse resultado deve ser dividido por 11. Se o resto da divisão for menor que 2, o DC1 será 0, caso contrário, o DC1 será 11 subtraído do resto obtido.
* DC2: Deve ser repetido o processo anterior considerando agora um número de

dez dígitos formado pelos nove dígitos do CPF à esquerda e o DC1 como sendo o dígito mais à direita do número. Ou seja, o DC1 deverá ser multiplicado pelo peso 2 e os demais dígitos com por um peso variando de 3 até 11.

1. Escreva uma função em C que recebe como parâmetros um vetor de inteiros representando os nove dígitos principais de um CPF (com o dígito mais à esquerda na primeira posição do vetor e assim sucessivamente), e calcula os valores de DC1 e DC2, devolvendo o resultado obtido através de dois ponteiros para inteiros, conforme o protótipo:

void controle(int \*cpf9d, int \*dc1, int \*dc2);

1. Escreva uma função em C que recebe como parâmetros um vetor de inteiros representando os onze dígitos do CPF (com o dígito mais à esquerda na primeira posição do vetor, assim sucessivamente, e DC1 e DC2 na penúltima e última posição, respectivamente), retornando 0 se os valores de DC1 e DC2 estão corretos ou 1, caso contrário. Esta função deve utilizar a função desenvolvida no item “a” e tem como protótipo:

int verifica(int \*cpf11d);

1. Utilizando a função desenvolvida no item “b”, escreva um programa que lê do teclado os 11 dígitos do CPF do usuário (fornecidos um de cada vez, da esquerda para a direita) e ao final imprime “CORRETO” ou “INCORRETO”, informando ao usuário se o DC digitado está correto ou não.
2. Implemente uma rotina recursiva para calcular o tamanho de uma lista estática.
3. Critique a seguinte versão da função busca:

int k = 0;

while (k < n && v[k] != x) k += 1;

if (v[k] == x) return k;

else return -1;

1. Critique a seguinte versão da função busca:

int sol;

for (k = 0; k < n; ++k) {

if (v[k] == x) sol = k;

else sol = -1; }

return sol;

1. Critique a seguinte função recursiva. O autor afirma que ela decide se x está em v[0..n-1].

int busc (int x, int n, int v[]) {

if (v[n-1] == x) return 1;

else return busc (x, n-1, v);

}

1. Que acontece se trocarmos "v[j-1] = v[j]" por "v[j] = v[j+1]" no código da função remover2?
2. Discuta a seguinte versão de remover2:

int remove\_r2 (int k, int n, int v[]) {

if (k < n-1) {

remove\_r2 (k, n-1, v);

v[n-2] = v[n-1];

}

return n - 1;

}

1. Refaça todo o problema da remoção sob condições mais gerais: Suponha que a parte relevante do vetor v é v[ini..fim-1]; para remove v[k], puxe v[k+1..fim-1] para a esquerda ou empurre v[ini..k-1] para a direita, dependendo de qual das alternativas seja mais "barata".
2. Refaça todo o problema da inserção sob condições mais gerais: Suponha que a parte relevante do vetor v é v[ini..fim-1]; para inserir x entre v[k-1] e v[k] você tem duas opções: empurrar v[k..fim-1] para a direita ou puxar v[ini..k-1] para a esquerda; escolha a opção mais apropriada.
3. Critique a seguinte função. Ela promete eliminar os zeros de v[0..n-1], deixar o resultado em v[0..m-1] e devolver m.

int tira0 (int n, int v[]) {

int i, k, z = 0;

for (i = 0; i < n - z; ++i) {

if (v[i] == 0) {

z += 1;

for (k = i; k < n - z; ++k) v[k] = v[k+1];

--i;

}

}

return n - z;

}