Estruturas de Dados – 2011/1
Prof[a] Patrícia Dockhorn Costa

Lista de exercícios 6

1) Dada uma árvore binária (campos ESQ, CHAVE, VALOR, DIR), escreva algoritmos para:

a) determinar quantos nós tem VALOR superior a um valor V (parâmetro);
b) imprimir os conteúdos (CHAVE e VALOR) das folhas da árvore;
c) alterar o campo VALOR do nó cuja chave é CH (parâmetro), para um novo valor VNOVO (outro parâmetro);
d) determinar a soma dos VALORes contidos em todos os nós;
e) determinar se a árvore é simétrica, (a) somente quanto à estrutura e (b) quanto à estrutura e ao conteúdo (CHAVE e VALOR) dos nós.

2) Considere uma árvore binária cujos nós são formados por:

NOME	CÓDIGO	DATA	PRODUTO	VALOR	ESQ	DIR

A raiz da árvore está no apontador RAIZ. Escreva algoritmos para:

a) imprimir os dados dos nós de datas posteriores a 05/09/07;
b) eliminar o nó de código COD.

3) Considere um TAD árvore binária genérico. Pede-se:

a) As declarações de tipos da árvore binária genérica;
b) Escreva a função genérica percorre (pré-ordem). A função percorre permite que cada nó da árvore binária genérica seja visitado. Lembre-se que a função percorre chama a função de callback para cada nó visitado;
c) Escreva a função de callback Imprime (imprime o nó visitado), que deve ser implementada pelo cliente do TAD genérico.

4) Explique os mecanismos de tratamento de colisão em tabelas "hash".

5) Considere um cadastro de alunos armazenado em uma tabela hash que usa listas encadeadas para o tratamento de colisões, isto é, a tabela é implementada como um vetor de listas encadeadas. Considerando as seguintes declarações relativas à tabela hash de alunos:

```
struct aluno {
    char nome[81];
    int matricula;
    float cr;
    struct aluno* prox;
};
typedef struct aluno Aluno;

#define N 127
typedef Aluno* Hash[N];
```

Implemente:

a) Uma função de hash que, dado o nome de um aluno, retorne o índice na tabela. A função deve seguir o protótipo:

```
int  hash (char* nome);
```

b) Uma função que insira um novo aluno na tabela hash. Para simplificar, considere que o aluno nunca estará presente na tabela. A função deve receber como parâmetros a tabela hash, o nome do aluno, sua matrícula e seu CR, e deve retornar um ponteiro para a estrutura criada para o aluno, de acordo com o seguinte protótipo:

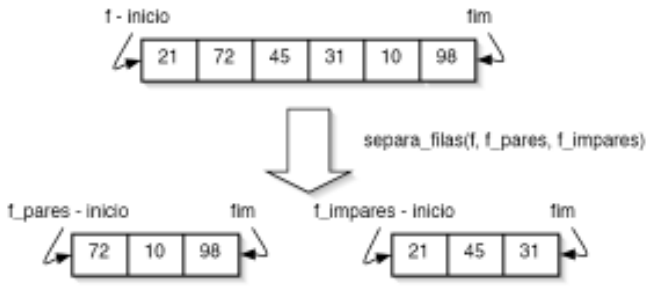
```
Aluno* insere (Hash tab, char* nome, int mat, float cr);
```

6) Considere uma árvore com número variável de filhos, usando a implementação que vimos em sala de aula. Implemente uma função que, dada uma árvore com número variável de filhos, retorne a quantidade de nós que guardam números pares. Essa funçao deve obedecer o protótipo: **int pares (ArvVar* a);**

7) Considere a existência de um tipo abstrato Fila de números inteiros, cuja interface está definida no arquivo fila.h da seguinte forma:

```
typedef struct fila Fila;
Fila* cria(void);
void insere (Fila* f, int v);
int retira (Fila* f);
int vazia (Fila* f);
void libera (Fila* f);
```

Sem conhecer a representação interna desse tipo abstrato Fila e usando apenas as funções declaradas no arquivo fila.h, implemente uma função que receba três filas, f, f_impares e f_pares, e separe todos os valores guardados em f de tal forma que os valores pares são movidos para a fila f_pares e os valores ímpares para f_impares, conforme ilustrado a seguir:



Note que, ao final dessa função, a fila f vai estar vazia. Essa função deve obedecer o protótipo:

void separa_filas (Fila* f, Fila* f_pares, Fila* f_impares);