Algoritmos e Estruturas de Dados III
2016 – 1
Descrição do Segundo Trabalho Prático

1 – Objetivo
O objetivo deste trabalho é implementar um sistema para manipulação de registros em um arquivo, indexado por meio de uma Árvore-B.

2 – Descrição
O sistema consistirá de 5 modulos: Interface, ArquivoIndexado, Arquivo, Registro e ArvoreB.

2.1 – O Módulo Registro
O módulo Registro é implementado pelos arquivos `Registro.h` e `Registro.c`.
Este módulo contém a definição do tipo de registro com o qual o sistema trabalha. A entidade a qual este registro representa deverá ser definida pelo próprio grupo (por ex.: um produto).
Cada registro deverá possuir pelo menos três atributos (por ex: nome, preço, código). Um destes atributos deverá ser escolhido como chave. Além destes três atributos, deverá haver um atributo adicional, que indica se o registro foi removido ou não do arquivo.
A chave deverá ter tipo `Chave`, a ser definido também neste módulo, por meio de um `typedef`. Deverão ser fornecidas funções para a comparação de chaves. Chaves só poderão ser comparadas por meio destas funções.
Os atributos do registro não poderão ser acessados diretamente por nenhum outro código. Para acessar os atributos de cada registro, deverão ser fornecidas funções.
Exemplo (`Registro.h`):
```
typedef int Chave;
int ComparaChave;
struct Registro
{
	char nome[100];
	double preco;
	Chave codigo;
};
struct Registro * CriaRegistro(char nome[100], double preco, Chave codigo);
char * getNome(struct Registro * P);
Chave getChave(struct Registro * P);
...
```

2.2 – O Módulo Arquivo
O módulo Arquivo é responsável por definir o tipo `Arquivo` e implementar funções para manter os registros em armazenamento secundário. Os registros serão mantidos em um arquivo binário.
Um *exemplo* de definição para o tipo arquivo é o seguinte:
```
struct Arquivo
{
	FILE * arq;
	char * nomeArq;
	int numRegistros;
};
```

Este módulo deve obrigatoriamente implementar as seguintes funções:

- `struct Arquivo * CriaArquivo(char * nome)`: Se existir um arquivo com o nome passado como parâmetro, esta função abre este arquivo, senão, a função cria um novo arquivo com este nome.
- `int Insere(struct Arquivo * A, struct Registro * R)`: Esta função insere o

registro R no fim do arquivo A e retorna a posição relativa do local onde o registro foi inserido.

- `int NumRegistros(struct Arquivo * A)`: Esta função retorna o número de registros atualmente no arquivo A.
- `struct Registro * Le(struct Arquivo *A, int p)`: Esta função retorna o registro armazenado na posição relativa p do arquivo A se `p < NumRegistros(A)`, retorna NULL, caso contrário.
- `void Escreve(struct Arquivo *A, struct Registro * R, int p)`: Esta função escreve o registro R, passado como parâmetro, na posição p do arquivo A, se `p < NumRegistros(A)`. Se `p > NumRegistros(A)`, então a função na faz nada. Esta função será útil para a atualização de registros no arquivo.

2.3 – O Módulo ArvoreB

O módulo ArvoreB implementa uma árvore-B em memória primária. A implementação deverá ser baseada na implementação fornecida pelo professor.

Cada nó da árvore armazenará um conjunto de chaves. Associado a cada chave, haverá um valor inteiro p.

Uma possível declaração para o tipo ArvoreB é a seguinte:

```
#define ORDEM 4
struct No
{
	Chave chaves[ORDEM-1];
	int valoresAssociados[ORDEM-1];
	struct No *Filhos[ORDEM];
	int numElementos;
};
struct ArvoreB
{
	struct No *raiz;
};
```

O módulo deverá fornecer as seguintes funções:

`struct ArvoreB * CriaArvoreB()`: Esta função cria e retorna uma árvore-B vazia.

`void Insere(struct ArvoreB * A, Chave chave, int p)`: Esta função insere na árvore a chave e o valor associado p, passados como parâmetro.

`int Remove(struct ArvoreB * A, Chave chave)`: Esta função remove chave da árvore e retorna o seu valor associado p.

`int Busca(struct ArvoreB * A, Chave chave)`: Esta função busca por chave na árvore, e retorna o seu valor associado p.

2.4 – O Módulo ArquivoIndexado

O módulo ArquivoIndexado é responsável por coordenar a integração do arquivo, em memória secundária, e seu índice, em memória primária. Para manter os registros em um arquivo, este módulo utiliza o módulo Arquivo. Para indexar, este módulo utiliza o módulo ArvoreB.

Uma possível definição para o tipo ArquivoIndexado é:

```
struct ArquivoIndexado
{
	struct Arquivo * arquivo;
	struct ArvoreB * indice;
};
```

O módulo deve implementar as seguintes funções:

`struct ArquivoIndexado * CriaArquivoIndexado(char * nome)`: Esta função deverá inicializar o arquivo indexado. A função deverá chamar a função `CriaArquivo` do módulo Arquivo, para abrir o arquivo com o nome correspondente ou criar um novo arquivo, caso não exista. Após isto, a função deverá percorrer os registros existentes no arquivo, por meio das funções

`NumRegistros` e Le do módulo Arquivo. Para cada registro existente que não tenha sido removido previamente, a função deverá inserir na árvore-B sua chave e sua posição. A posição relativa de cada registro será armazenada no atributo `p` associado à chave.
`void Insere(struct ArquivoIndexado * A, struct Registro * R)`: Esta função insere um novo registro no arquivo e sua referência no índice. Para isto, a função executa a função `Insere` do módulo arquivo, aquela função inserirá o registro no arquivo e retornará sua posição. Após isto, a função `Insere` do módulo ArvoreB deverá ser executada, para inserir na árvore-B a chave do novo registro e sua posição no arquivo.
`struct Registro * Buscar(struct ArquivoIndexado * A, Chave chave)`: Esta função busca pelo registro com a chave passada como parâmetro, caso existir. Para isto, a função deve executar a função `Busca` da árvore-B, que retornará a posição do registro no arquivo. Após isto, a função deve executar a função Le do módulo Arquivo, para recuperar o registro com a posição correspondente.
`struct Registro * Remover(struct ArquivoIndexado * A, Chave chave)`: Esta função remove do índice a referência ao registro com a chave passada como parâmetro. Para isto, a função executa o método `Remove` do módulo ArvoreB. Aquela função removerá a referência ao registro da árvore, e retornar sua posição. Esta posição deverá ser usada como parâmetro do método `Le` do módulo Arquivo, para recuperar o registro e retorná-lo ao fim da chamada. O registro lido deverá ser marcado como removido e escrito novamente no arquivo, na mesma posição. Observe que esta função não removerá de fato o registro do arquivo, apenas da árvore. O registro continuará no arquivo, mas estará marcado como removido, e será ignorado em execuções posteriores do programa.

2.4 – O Módulo Interface

O módulo Interface faz a conexão do usuário com o sistema. Este módulo deve fornecer as seguintes operações:
`Inserir`: Permite que o usuário insira um novo registro no sistema
`Remover`: Permite que o usuário remova um registro do sistema, dada a sua chave.
`Buscar`: Permite que o usuário procure por um registro no sistema, dada a sua chave. Se o registro existir, suas informações são exibidas. Senão, é exibida a mensagem "Registro não encontrado".
`Alterar`: Permite que o usuário altere o conteúdo de um registro, se existir.

3 – Entrega

Deverão ser entregues os arquivos-fonte do sistema, acompanhados de um relatório. O relatório deverá explicar como compilar e como executar o seu programa em Linux. Além disso, o relatório deverá explicar, resumidamente, para cada uma das opções do menu (inserir, remover, buscar, alterar), as ações executadas pelo sistema quando tal opção for selecionada.

O trabalho deverá ser implementado utilizando a linguagem de programação C e ser passível de compilação em Linux.
O trabalho deverá ser feito em grupos de no máximo 3 alunos.
A data de entrega é 07/08/2016, até as 23:55.