ACH2024 ALGORITMOS E ESTRUTURAS DE DADOS II

Semestre 2021-1 - Exercício prático 2 – inserção em árvore B– t.02

Estagiário PAE: Wesley Ramos dos Santos (wesley.ramos.santos@usp.br)

Descrição do EP: A partir do modelo disponibilizado no e-disciplinas (*ep2-modelo.cpp*), implementar as seguintes operações em uma árvore B armazenada em arquivo.

Antes de começar: certifique-se de ter estudado o conteúdo das aulas 12 (índices não residentes) e 13 (inserção em árvore B).

1. O objetivo do trabalho é implementar de forma correta e completa a função **inserir** em um arquivo de registros representando páginas de uma árvore B inicialmente vazia (ver *typedef* do código exemplo fornecido), e demais funções auxiliares para realização da operação solicitada. Não modifique as definições de *typedef* do código, ou seu programa será invalidado.
2. A árvore armazena chaves do tipo inteiro, e possui ordem m=3, ou seja, 3 links por página, e portanto cada página pode ter até 2 chaves numeradas como 1 e 2. Cada página é formada por um vetor de itens do tipo CHLINK que corresponde a um par de chave e link (direito exceto para o item 0, que é na verdade um link esquerdo). Um campo contador é incluído para armazenar a quantidade de chaves efetivamente armazenadas em cada página.

   item[0]: chave[0] desprezada, e linkdir[0] usado como o link esquerdo da chave[1]

   item[1]: chave[1] e linkdir[1]

   Item[2]: chave[2] e linkdir[2].
3. A função recebe como entrada um nome de arquivo a ser aberto representando uma árvore B vazia ou não, o número do registro raiz (**-1 para árvore vazia**, e passado por referência porque pode mudar), e uma chave a ser inserida se possível.

   void inserir(char *nomearq, int* raiz, int ch)

4. **Os registros do arquivo (e portanto os números de páginas) devem ser contados a partir de zero, e ao criar a primeira página, ela normalmente vai ocupar a posição zero do arquivo. Em outras palavras, se raiz == -1 a árvore não existe, e se raiz > -1 então a árvore começa no registro indicado.**
5. **Pelo mesmo motivo, os links dentro de cada página devem apontar para o número da página filho em questão, ou apontar para -1 caso estejam na posição folha.**
6. Note que, por simplicidade, essa representação contempla apenas as chaves do índice e os links para as outras páginas da estrutura, não havendo assim o campo com o endereço para o arquivo da aplicação que é o verdadeiro objetivo do índice. Em outras palavras, neste EP não será tratado do acesso ao arquivo da aplicação propriamente dito, mas apenas da organização da estrutura do tipo árvore B.
7. O arquivo da árvore B é formado por registros do tipo PAGINA conforme indicado no código exemplo, e é portanto um arquivo binário contendo registros de tamanho fixo a ser manipulado exclusivamente com *fread*, *fwrite* e *fseek*.
8. A inserção deve certificar-se de que a chave não existe, e inseri-la na posição correta se for o caso. Além disso, deve tratar casos de *overflow* implementando a operação de divisão (*split*) e, consequentemente, realizar a promoção da chave excedente aos níveis superiores (em último caso podendo levar à criação de uma nova raiz).
9. Lembre-se: como qualquer árvore de busca, a árvore B é ordenada e não admite chaves repetidas. Se a chave a ser inserida já existir o algoritmo não deve fazer nada.
10. Não use variáveis globais. A função implementada deve definir localmente todas as variáveis e estruturas auxiliares, ou chamar funções auxiliares que o façam também em um escopo local.

11. Não exiba nenhuma mensagem na tela, nem solicite que o usuário pressione nenhuma tecla etc. Apenas implemente a função solicitada.
12. A função *main*() serve apenas para seus testes particulares, e não precisa ser entregue. Caso você prefira mantê-la no corpo do programa, pede-se apenas que *main*() seja a última função do programa, ou seja, que não haja nenhum código abaixo dela.
13. Seu programa será corrigido de forma *automática*, e por isso você não pode alterar as assinaturas da função solicitada, nem os tipos de dados ou especificações (*typedef*) do modelo fornecido.
14. O EP deve ser desenvolvido individualmente. Não tente emprestar sua implementação para outros colegas, em copiar deles, pois isso invalida o trabalho de **todos** os envolvidos.
15. 0 programa deve ser compilável no ambiente Windows com Codeblocks 13.12 ou superior. Será aplicado um desconto de até 30% na nota do EP caso ele não seja imediatamente compilável nesta configuração.

O que/como entregar:

- A entrega será via upload no sistema e-disciplinas antes da data estipulada.
- Entregue apenas um arquivo texto com o código da função principal e funções auxiliares que ela invoca.
- A extensão do arquivo deve ser .cpp - **favor não compactar**.
- Preencha as funções nroUSP do código exemplo disponível para que você seja identificado.
- Na primeira linha do código, escreva um comentário "//" com seu nome.

Prazos etc.:

O EP deve ser depositado no prazo definido na atividade cadastrada no sistema. Não serão aceitos EPs entregues depois do prazo, independentemente do motivo. Entregas no último dia são assim por conta e risco do aluno, e nenhum tipo de imprevisto de última hora (e.g., problemas de saúde, indisponibilidade de rede etc.) pode ser usado como justificativa para o atraso. O EP é uma atividade para ser desenvolvida ao longo de várias semanas, não nos últimos dias antes da entrega.

É responsabilidade do aluno que fez o *upload* do arquivo verificar se o mesmo foi corretamente recebido pelo sistema. Atrasos/falhas na submissão invalidam o trabalho realizado. Após o *upload*, verifique se você consegue abrir o arquivo depositado, e certifique-se de que é a versão correta do programa e que não está corrompido.

Critérios de avaliação:

A função será testada com uma série de chamadas repetidas e consecutivas, com diversas chaves de entrada (novas ou repetidas). É assim importante assegurar que o seu programa funciona desta forma (por exemplo, chamando-o dentro de um laço *for*), e não apenas para um teste individual. Um teste é considerado correto se o resultado da soma for exatamente como o esperado, ou incorreto em caso contrário. Erros de alocação de memória ou compilação invalidam o teste, assim como a ausência de funções auxiliares necessárias para a execução do programa.

Este EP deve ser desenvolvido obrigatoriamente por *todos* os alunos de AED2. Sua nota é parte integrante da avaliação e *não é* passível de substituição.