BCC202 – Estruturas de Dados I

Departamento de Computação - Universidade Federal de Ouro Preto - MG
Professor: **Pedro Silva** (`www.decom.ufop.br/`)

ÁRVORE BINÁRIA DE PESQUISA

- **Submissão com data e hora de entrega disponíveis na plataforma da disciplina**.

- Procedimento para a entrega:.

1. Siga atentamente quanto ao formato da entrada e saída de seu programa, exemplificados no enunciado.
2. Durante a correção, os programas serão submetidos a vários casos de testes, com características variadas.
3. A avaliação considerará o tempo de execução e o percentual de respostas corretas.
4. Eventualmente, serão realizadas entrevistas sobre as práticas para complementar a avaliação.
5. Considere que os dados serão fornecidos pela entrada padrão. Não utilize abertura de arquivos pelo seu programa. Se necessário, utilize o redirecionamento de entrada.
6. Os códigos fonte serão submetidos a uma ferramenta de detecção de plágios em software.
7. Códigos cuja autoria não seja do aluno, com alto nível de similaridade em relação a outros trabalhos, ou que não puder ser explicado, acarretará na perda da nota.
8. Códigos ou funções prontas específicos de algoritmos para solução dos problemas elencados não são aceitos.
9. Não serão considerados algoritmos parcialmente implementados.

- Bom trabalho!

Um sistema de gestão de preços de sementes

Você foi contratado por uma empresa de tecnologia agrícola chamada **AgroTech**, que está desenvolvendo um sistema para ajudar fazendeiros a otimizar suas colheitas. A **AgroTech** precisa de um algoritmo eficiente para gerenciar os preços de venda das sementes de diversas culturas em diferentes fazendas. Esses preços desconsideram os centavos.

Os preços das sementes são atualizados frequentemente, e o sistema deve permitir a inserção de novos preços e a busca por todos os produtos que estão em um intervalo de preços específicos (deve-se ser mostrado qual cultura e o nome da fazenda). Além disso, o sistema deve ser capaz de organizar e buscar esses preços rapidamente, mesmo com grandes quantidades de dados. Outro ponto importante é que foi feita uma análise e que diferentes culturas em diferentes fazendas podem ter o mesmo preço, todavia, isso ocorrerá **dez** vezes para um mesmo preço.

Para garantir um desempenho eficiente na manipulação desses dados, você decidiu implementar um TAD de Árvore Binária de Pesquisa para organizar os preços das sementes de maneira ordenada. Esse TAD permitirá ao sistema: (1) inserir os preços e o nome da cultura, além do nome da fazenda responsável; (2) imprimir a cultura e o nome das fazendas que tem um preço em um intervalo específico.

Considerando o cenário exposto, o seu objetivo é desenvolver um sistema em C que usa um TAD para organizar e gerenciar os preços de sementes. Você irá implementar as operações de inserção e busca/impressão.

Considerações

O código-fonte deve ser modularizado corretamente conforme os arquivos de protótipo fornecidos. O problema deve ser resolvido pela implementação de *Árvores Binárias de Pesquisa* para armazenar e manipular os dados.

Não se sabe a quantidade de produtos que serão inseridos, só que o mesmo preço pode se repetir no máximo **dez vezes**. Cada caso de teste deve ser resolvido em até 1 segundo.

- Não altere o nome dos arquivos.
- O arquivo `.zip` deve conter na sua raiz somente os arquivos-fonte.
- Há vários casos de teste. Você terá acesso (entrada e saída) de casos específicos para realizar os seus testes.

Especificação da Entrada e da saída

A entrada consiste em um único caso de teste. Cada linha contém um preço (número inteiro), o nome da cultura (até 20 caracteres) e o nome da fazenda responsável (até 50 caracteres). A leitura das linhas deve continuar até que um preço igual a zero seja encontrado. Quando isso acontecer, devem ser lidos mais dois números inteiros: a e b, sendo que a é sempre menor ou igual a b. Esses números representam o intervalo de itens que devem ser impressos.

A saída deve mostrar todos os itens com um preço dentro do intervalo especificado. A impressão deve ser feita nas ordens: *InOrder*, *PreOrder* e *PosOrder*.

A entrada é composta de um único caso de teste. Cada linha consiste em um preço (inteiro), o nome da cultura (20 caracteres) e o nome da fazenda responsável (50 caracteres). A leitura deve ocorrer até encontrar um preço igual a zero, uma vez que ele foi encontrado, deve-se ler mais dois inteiros: a e b, onde a sempre é menor ou igual a b. Esses dois inteiros representam o início e o fim inclusos de um intervalo onde todos os itens dentro desse intervalo devem ser impressos.

A sua saída é a impressão de todos os itens no intervalo especificado. A impressão deve ser feita usando o caminhamento *InOrder*, *PreOrder* e *PosOrder* respectivamente.

Exemplo de entrada e saída:

Entrada	**Saída**

`50 milho sitioalegria`
`30 trigo sitiovaledapaz`
`70 soja fazendadaluz`
`20 cevada fazendaazul`
`50 cafe sitiocabecao`
`60 arroz fazendacross`
`80 algodao sitiogalo`
`0`
`10 70`
	`InOrder:`
	`20 ( [cevada|fazendaazul] )`
	`30 ( [trigo|sitiovaledapaz] )`
	`50 ( [milho|sitioalegria] [cafe|sitiocabecao] )`
	`60 ( [arroz|fazendacross] )`
	`70 ( [soja|fazendadaluz] )`
	
	`PreOrder:`
	`20 ( [cevada|fazendaazul] )`
	`30 ( [trigo|sitiovaledapaz] )`
	`60 ( [arroz|fazendacross] )`
	`70 ( [soja|fazendadaluz] )`
	`50 ( [milho|sitioalegria] [cafe|sitiocabecao] )`
	
	`PosOrder:`
	`50 ( [milho|sitioalegria] [cafe|sitiocabecao] )`
	`30 ( [trigo|sitiovaledapaz] )`
	`20 ( [cevada|fazendaazul] )`
	`70 ( [soja|fazendadaluz] )`
	`60 ( [arroz|fazendacross] )`

Diretivas de Compilação

```
$ gcc -c arvore.c -Wall
$ gcc -c pratica.c -Wall
$ gcc arvore.o pratica.o -o exe
```

Avaliação de *leaks* de memória

Uma forma de avaliar se não há *leaks* de memória é usando a ferramenta valgrind. Um exemplo de uso é:

```
gcc -g -o exe *.c -Wall; valgrind --leak-check=yes -s ./exe < casoteste.in
```

Espera-se uma saída com o fim semelhante a:

```
==38409== ERROR SUMMARY: 0 errors from 0 contexts (suppressed:  0 from 0)
```

Para instalar no Linux, basta usar: `sudo apt install valgrind`.