Laboratório de Programação II

Árvore Binária

Universidade Federal de Juiz de Fora
Departamento de Ciência da Computação

Aula de Hoje

- TAD Árvore Binária
  - Revisão
  - Criação *bottom-up* de uma árvore binária
  - Criação *top-down* de uma árvore binária
  - Exercícios

TAD Nó

- TAD Nó para árvore binária.

```
class NoArv
{
  public:
    NoArv()                  { };
    ~NoArv()                 { };
    void setEsq(NoArv* p)    { esq = p; };
    void setInfo(int val)    { info = val; };
    void setDir(NoArv* p)    { dir = p; };
    NoArv* getEsq()          { return esq; };
    int getInfo()            { return info; }
    NoArv* getDir()          { return dir; };

  private:
    NoArv *esq; // ponteiro para o filho a esquerda
    int info;   // informacao do No (int)
    NoArv *dir; // ponteiro para o filho a direita
};
```

TAD ArvBin

- TAD Nó para árvore binária.

```
class ArvBin
{
  public:
    ArvBin();
    ~ArvBin();
    int getRaiz();
    void cria(int x, ArvBin *sae, ArvBin *sad);
    void montaArvore();
    void insere(int x);
    bool vazia();
    void preOrdem();
  private:
    NoArv* raiz;
    NoArv* libera(NoArv *p);
    NoArv* auxMontaArvore();
    NoArv* auxInsere(NoArv *p, int x);
    void auxPreOrdem(NoArv *p);
};
```

TAD ArvBin

- Lembre-se, para algumas operações que utilizam um algoritmo recursivo para realizar alguma tarefa é preciso criar uma função auxiliar que recebe como parâmetro um ponteiro para um nó.
- Na primeira chamada a função passa a raiz.
- Em seguida a função auxiliar trabalha de forma recursiva para implementar o algoritmo desejado.

```
class ArvBin
{
  public:
    // ....
    void preOrdem();

  private:
    // ...
    void auxPreOrdem(NoArv *p);
};
```

Exercícios

1. Desenvolva uma operação para contar quantos o total de nós de uma árvore binária.

```
int ArvBin::contaNos();
```

2. Desenvolva uma operação para contar quantos nós são folhas em uma árvore binária.

```
int ArvBin::contaNosFolhas();
```

3. Desenvolva uma operação para determinar a altura de uma árvore binária.

```
int ArvBin::altura();
```

4. Desenvolva uma operação para calcular e retornar a quantidade de valores ímpares numa árvore binária.

```
int ArvBin::contaImpar();
```

Exercícios

5. Desenvolva uma operação para calcular a quantidade de valores ímpares armazenados nos nós folhas de uma árvore binária.

```
int ArvBin::contaFolhaImpar();
```

6. Desenvolva uma operação para imprimir todos os valores dos nós de um dado nível k da árvore binária.

```
void ArvBin::imprimeNivel(int k);
```

7. Desenvolva uma operação para calcular a média dos valores dos nós de um dado nível k.

```
float ArvBin::mediaNivel(int k);
```

8. Desenvolva uma operação para encontrar o menor (e o maior) valor de uma árvore binária.

```
int ArvBin::min();
int ArvBin::max();
```

Exercícios

9. Escreva uma operação que *inverte* a árvore binária, isto é, todo nó que tiver 2 filhos terá o seu valor do filho à esquerda trocado com o do filho à direita.

```
void ArvBin::inverte();
void ArvBin::auxInverte(NoArv *p);
```

10. Desenvolva uma operação para retornar o valor do nó mais à esquerda da árvore binária. Em seguida, faça outra operação para retornar o valor do nó mais à direita da árvore.

```
int ArvBin::noMaisEsquerda();
int ArvBin::noMaisDireita();
```

Exercícios

11. Desenvolva operações para, dado um ponteiro para um nó qualquer, encontrar o menor (e o maior) valor na árvore.

```
int ArvBin::minSubArvore(NoArv *p);
int ArvBin::maxSubArvore(NoArv *p);
```

12. Faça uma operação para verificar se uma árvore binária é uma árvore binária de busca (ABB). Dica: utilize as funções definidas no Exercício 8 de forma auxiliar. Por exemplo, se o maior valor da sub-árvore à esquerda for maior do que o valor da raiz, então não é ABB. Faça um teste similar para a sub-árvore à direita.

```
bool ArvBin::ehABB();
bool ArvBin::auxEhABB(NoArv *p);
```