INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO PIAUÍ

CAMPUS TERESINA-CENTRAL

DIRETORIA DE ENSINO

Estrutura de Dados II – Ordem de percurso - Continução
- Aula 3 -

Professora: Elanne Cristina O. dos Santos

elannecristina.santos@gmail.com
elannecristina.santos@ifpi.edu.br

Solução sem recursão /usando pilha (fig. 6.15 pag 201):

```
void preOrder(){
    stack<No*> pilha;
    No *p = raiz;
    string v;
    if (p!=0){
        pilha.push(p);
        while (!pilha.empty()) {
            p=pilha.top();
            cout<<pilha.top()->nome<<endl;
            pilha.pop();
            if (p->right !=0)
                pilha.push(p->right);
            if (p->left != 0)
                pilha.push(p->left);
        }
    }
}
```

Pré-ordem : Solução sem recursão /usando pilha

- Segundo o autor (livro pag. 201):
  - "O algoritmo é duas vezes tão grande quanto a versão usando recursão, mas ainda é pequeno e legível."
  - "No entanto usa pesadamente a pilha. Em consequência funções de suporte são necessárias para processar a pilha, logo a implementação total não é tão pequena."

Pré-ordem : Solução sem recursão /usando pilha

- Segundo o autor (livro pag. 201):
  - “Embora duas chamadas recursivas sejam omitidas, existem agora até 4 chamadas por iteração do laço while: até duas chamadas de push(), uma chamada de pop() e uma chamada de visit(). É difícil considerar isso como melhoria de eficiência. “

OBSERVAÇÃO: o **visit()** citado pelo autor foi substituído na versão apresentada nesse slide pela linha:

cout << pilha.top()->nome<<endl;

Atividade

- Implementar pré-ordem, pos-ordem e in-ordem usando uma pilha.

- Verifique o tempo de execução de cada um dos algoritmos nas duas versões e confira se realmente o uso da pilha auxiliar ao invés do uso da recursão otimizou o algoritmo gastando menos tempo de processamento.
  - Existe melhoria de eficiência do algoritmo?