**Trabalho 02: Aplicações de Algoritmos em Árvores**

SSC0300 – Linguagens de Programação e Aplicações

**Nome:** Felipe Augusto Senger Lopes de Souza

**No USP:** 5513317

**Introdução:** Este projeto contém os 3 exercícios do 2º Trabalho da disciplina de Linguagens de Programação e Aplicações. Eles são programas em C que utilizam os conceitos de Algoritmos em Árvores para resolver os problemas propostos.

**Descrição do Projeto:** Todos os programas foram desenvolvidos no Windows 7 64 bits, no programa Dev C++. O compilador utilizado foi o default do Dev C++, v5.10, nas configurações TDM-GCC 4.8.1 64-bit Release. Todas as bibliotecas usadas foram as consideradas padrão do Dev C++. A compilação não requer nenhum parâmetro especial e os programas foram compilados usando o botão de compilação do Dev C++.

Segue a descrição de como os programas funcionam e são executados:

**Exercício 1 – Operações Básicas em Árvores:**

O programa recebe a entrada de uma quantidade n de números inteiros e os armazena em uma Árvore Binária de Busca (ABB). Além disso, ele permite a inserção de novos números, a remoção de números (inclusive da raiz), a busca de números, sua impressão Em Ordem, Pré Ordem e Pós Ordem e sua impressão em Labelled Bracketing.

Para que tudo isso seja feito, foram criadas diferentes funções e o programa conta com um menu, que pergunta ao usuário o que ele deseja fazer, e chama as determinadas funções. Para inserir números, o programa chama a função de inserção para cada número, criando os nós um por um (alocando a memória um por um com o uso do *malloc*). Para remover números, existem duas funções, devido ao caso específico da remoção da raiz, ou seja, existe uma função para remover a raiz e outra para remover outros números.

Quanto à organização da Árvore, o programa organiza o número a partir de uma raiz. Números menores são inseridos à esquerda de números maiores, e números maiores à direita, e assim a árvore fica ordenada (números iguais são colocados à esquerda, para que o programa consiga trabalhar também com números iguais). É importante ressaltar que por se tratar de uma árvore binária, cada nó possui no máximo dois ramos. Para montar a árvore é preciso dizer ao programa quantos números serão inseridos, e depois os números a serem organizados. Notar que o primeiro número será a raiz, e os demais serão organizados da forma descrita acima.

**Exemplo:** Caso o usuário entre com a seguinte sequência

**7**

**45**

**12**

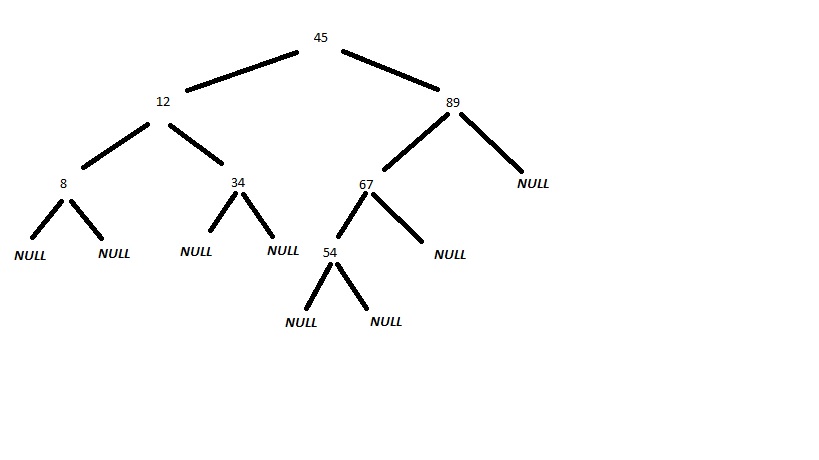
**89**

**67**

**34**

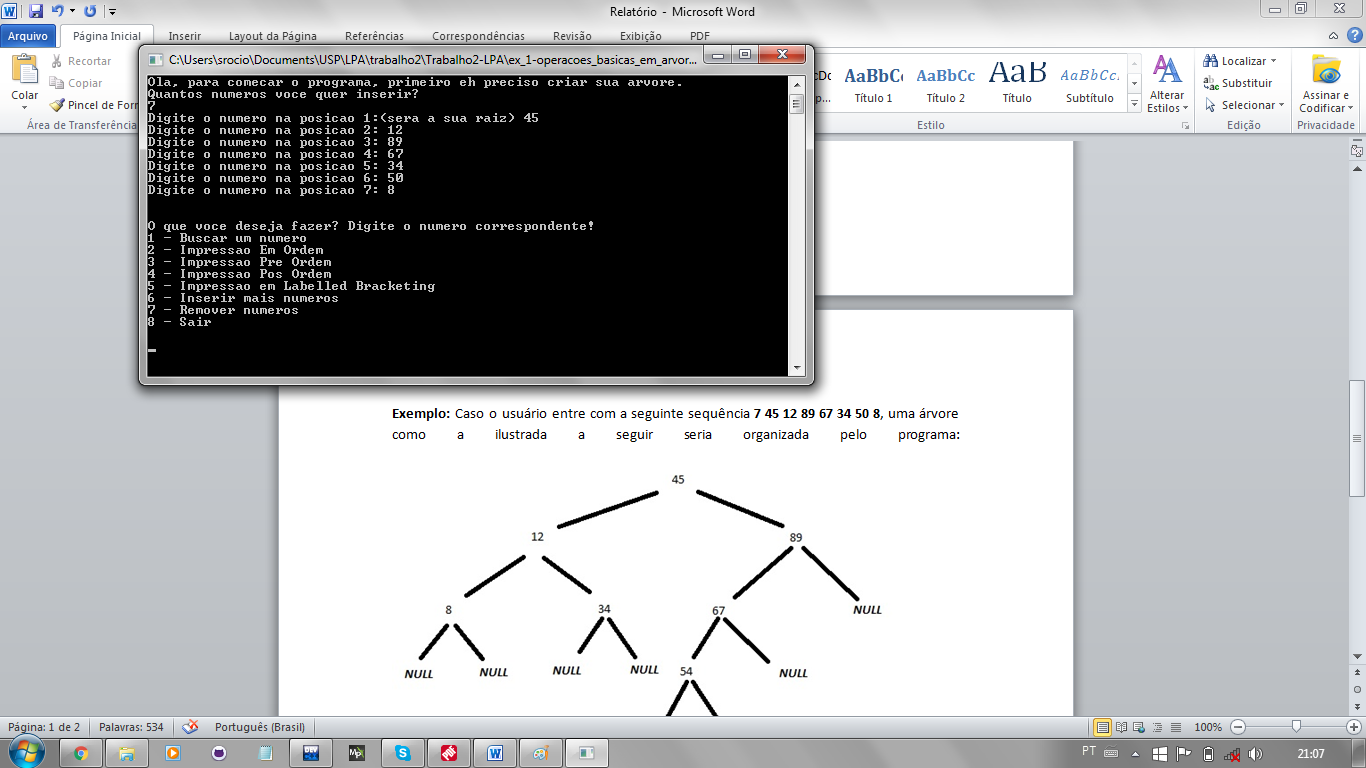
**50**

**8**

Uma árvore como a ilustrada a seguir seria organizada pelo programa:

Árvore-exemplo - exercício 1

**Tutorial de uso:** O programa deve ser compilado nas configurações citadas acima, que são as consideradas “default” do Dev C++ (versão 5.10). O primeiro número a ser entrado é a quantidade n de números que o usuário quer inserir na árvore, em seguida, deve-se inserir os n números, sabendo que o primeiro será a raiz da árvore. Feita a inserção, um menu irá aparecer, para que o usuário possa escolher o que deseja que o programa faça:



Menu - exercício 1

Basta digitar o número e apertar ‘Enter’, por exemplo, se deseja obter a impressão da árvore em “Labelled Bracketing”, deve-se digitar **5** e depois teclar ‘Enter’.

**Entrada exemplo: 7 45 12 89 67 34 50 8**

**Exercício 2 – Árvore genealógica:**

O programa recebe a entrada de uma quantidade n de tuplas, ou seja, um grupo de nomes da família, na ordem “filho mãe pai”, e os armazena numa árvore binária (fazendo a consideração de que cada casal possui apenas 1 filho). Além disso, o programa permite a impressão de cada geração, impressão dos antepassados de um dado indivíduo, impressão da árvore em “labelled bracketing” e o cálculo do grau equivalente de parentesco entre duas dadas pessoas, além de permitir a inserção de mais membros na árvore.

Para que tudo isso seja feito, foram criadas diferentes funções e o programa conta com um menu, que pergunta ao usuário o que ele deseja fazer, e chama as determinadas funções. Para inserir as “tuplas”, o programa chama a função de inserção para cada tupla, criando os nós um por um (alocando a memória um por um com o uso do *malloc* para cada nome dado). Para fazer o controle do “nível” de cada indivíduo na árvore, cada pessoa possui um grau. O filho tem grau 0, os pais grau 1, avós grau 2, e assim por diante. Esse grau é utilizado para calcular o grau de parentesco entre duas pessoas (por meio de uma subtração entre os dois graus) e para imprimir as gerações da árvore. A função que imprime as gerações da arvore faz uso de um algoritmo de percorrimento da árvore Em Ordem, pois é necessário procurar na arvore toda as pessoas de cada grau, podendo imprimir as determinadas gerações. Também foi feita uma função recursiva para a impressão em “labelled bracketing”, e uma função *select,* muito utilizada na execução do programa, pois ela é capaz de localizar um dado nome na árvore, e retornar o seu endereço, para que ele possa ser utilizado em diferentes funções como a de inserção de números e a do cálculo do grau de parentesco equivalente.

Quanto à organização da árvore, o primeiro filho inserido é a raiz. Os dois ramos que saem a partir dele são seus pais, os dois ramos de cada pai os avós, e assim por diante. É preciso fornecer ao programa quantas “tuplas” serão entradas, cada uma contendo um filho e seus pais. É importante notar que a partir da segunda “tupla”, o nome do filho deve já pertencer à arvore, e portanto o usuário deve fornecer o nome do filho de quem se quer dar o nome dos pais, e depois o nome dos pais.

**Exemplo:** Para uma entrada como a seguinte

**4**

**Felipe Simone Paulo**

**Paulo Vera Ricardo**

**Simone Leda Silvio**

**Vera Sonia Nelson**

O programa irá organizar a árvore ilustrada a seguir:

OBS importante seguir ordens de entrada

OBS

No exercício 2, para atender ao modelo de entrada proposto, o programa apenas aceita nomes simples para cada pessoa, que são separados por espaço.

Não funciona com pessoas com o mesmo nome