# Trabalho da AA de Estrutura de dados 1

## Tema: Manipulação sobre conjuntos.

O trabalho é individual. Ao final do semestre, a atividade acadêmica será cumprida com: apresentação do trabalho, relatório e a entrega das implementações utilizadas.

A data de entrega é até **18 de dezembro de 2017**.

**Descrição do trabalho**: O aluno deverá implementar um programa que permita criar e manipular conjuntos, conforme as opções: criar conjunto; destruir conjunto; pesquisar um elemento em um conjunto; união de dois conjuntos; e, intersecção de dois conjuntos.

Para isso, o programa deve conter um *menu* (*menu* principal) com as seguintes opções:

1 - Criar conjunto.

2 - Destruir conjunto.

3 - Pesquisar um elemento em um conjunto.

4 - União de dois conjuntos.

5 - Intersecção de dois conjuntos.

6 - Exibir árvores.

7 - Exibir todos os elementos de um conjunto em ordem.

8 - Exibir altura da árvore/árvore balanceada.

F - Fim do programa.

## Funcionamento do programa

Cada opção do *menu* tem uma funcionalidade específica conforme descrito nas Subseções a seguir. Os elementos do conjunto são representados por uma variável composta com dois campos: campo índice e site, onde o índice é um inteiro **sem sinal** e o site é uma **string**. Para trabalhar string no C-ANSI, considere o CHAR[256]. Um exemplo de elemento do conjunto é mostrado pelo código[1] 1.

Código 1: Estrutura do elemento da lista

```
struct stElemento
{
    unsigned int indice;
    char site[256];
    void *proxElemento;
};
typedef struct stElemento tpElemento;
```

Para gerenciar os conjuntos, será necessário utilizar um lista de árvores de conjuntos. A lista é definida como $L$ para fins didáticos. A lista $L$ é uma lista composta por árvores $a_i$, onde elemento da lista $L$ tem um índice inteiro, um ponteiro para os elementos da árvore ($a_i$) e um ponteiro para a próxima árvore ($a_{i+1}$), conforme apresentado pelo código[2] 2.

[1] O código considera uma lista duplamente encadeada

[2] O código considera uma lista simplesmente encadeada

Código 2: Estrutura do elemento da lista

```
struct stPonteiroElemento
{
    int indice;                          //Indice da lista L
    void *prox;                          //Proximo elemento da lista L
    tpElemento *topo_lista_i;    //Ponteiro para o topo da lista l

};
typedef struct stPonteiroElemento tpPonteiroElemento;
```

Os elementos de um conjunto $i$ devem ser armazenados em uma árvore binária $a_i$. Cada árvore $a_i$ deve conter um identificador único e deve ser armazenada em uma lista $L$. Este identificador único deverá ser dado pelo programa e armazenado em uma variável do tipo inteiro sem sinal, conforme é ilustrado pelo código 2.

## Opção: 1 - Criar conjunto.

Esta opção permite ao usuário a criar um novo conjunto. Para criar um conjunto, o programa deve fornecer um inteiro único como índice da árvore $a_i$ e pedir para o usuário entrar com o site.

A cada novo elemento, o programa deverá gerar o índice único e crescente, iniciando por 1, para cada site digitado e inserir o elemento criado na árvore $a_i$ em ordem crescente do índice, ou seja, os elementos menores que o índice pai devem ser inseridos no ramo a esquerda da árvore e os maiores no ramo a direita . Após cada inserção, o programa deverá perguntar se o usuário deseja continuar inserindo um novo site na lista $l_i$ ou sair.

No caso de não desejar mais inserir elementos (opção sair), o programa deverá retornar ao *menu* principal e armazenar a lista $l_i$ na lista $L$. Caso contrário (continuar inserindo), o programa deverá pedir para o usuário digitar o site.

## Opção: 2 - Destruir conjunto.

Esta opção permite ao usuário destruir um conjunto existente. Para isto, o programa deve pedir ao usuário que forneça o índice do conjunto $a_i$.

A destruição do um conjunto $a_i$ envolve desalocar todos os elementos deste conjunto da memória do computador e atualizar os respectivos ponteiros.

No caso do usuário digitar um índice inválido, ou seja, um conjunto que não existe. O programa deverá apresentar uma mensagem de erro informando que o conjunto é inválido e aguardar que o usuário pressione [ENTER] para retornando para a opção em que é pedido ao usuário que forneça um índice de um conjunto $a_i$.

## Opção: 3 - Pesquisar um elemento em um conjunto.

Esta opção permite ao usuário consultar se um elemento existe ou não. Para isto, o programa deve pedir que o usuário informe o índice da árvore $a_i$ e o índice do elemento que deseja procurar. A pesquisa envolve selecionar uma árvore $a_i$ válida e pesquisar dentro desta lista o elemento informado pelo usuário.

Para o caso em que a pesquisa ache o elemento, o programa deve imprimir na tela o índice e o site do elemento e aguardar que o usuário digite [ENTER] para perguntar se deseja ou não fazer uma outra conjunto. Não sendo feita mais nenhuma pesquisa, o programa deverá retornar ao menu principal. Para o caso de uma nova consulta, o programa deve pedir novamente para o usuário informar o índice do conjunto $a_i$ e o elemento que deseja consultar.

No caso do índice inválido de uma árvore $a_i$, o programa deve exibir uma mensagem de error informando que a lista é inválida e, após a leitura da mensagem pelo usuário, onde o mesmo deve pressionar [ENTER]. O programa deverá pedir novamente que seja informe o índice da árvore $a_i$ e o índice do elemento.

## Opção: 4 - União de dois conjuntos.

Nesta opção, o programa deverá criar um novo conjunto $a_i$, atribuído a este conjunto um índice e armazenando esta lista $a_i$ em $L$. Para isto, o usuário deverá fornecer dois índices $a_i$ válidos. Onde o primeiro índice refere-se a um conjunto "A" e outro índice refere-se a um conjunto "B".

A operação de união consiste em criar um novo conjunto com todos os elementos do conjunto "A" e do conjunto "B". Onde o programa vai considerar o índice dos elementos para eliminar as repetições ou os elementos de interseção. O programa deverá permitir a união de dois conjuntos vazios.

No caso de um dos conjuntos ser inválido, o programa deverá informar o usuário qual conjunto é inválido e aguardar que o usuário pressione [ENTER] para continuar. E, pedir novamente que informe os dois conjuntos.

### Opção: 5 - Intersecção de dois conjuntos.

Nesta opção, o programa deverá criar um novo conjunto $l_i$, atribuído a este conjunto um índice e armazenando esta lista $l_i$ em $L$. Para isto, o usuário deverá fornecer dois índices $l_i$ válidos, onde o primeiro índice será referente a um conjunto "A" e outro índice para referenciar o conjunto "B".

A operação de intersecção consistem em criar um novo conjunto com os elementos que estão no conjunto "A" e também estão no conjunto "B". Onde o programa vai considerar o índice dos elementos. Na operação de interseção, o programa deverá permitir a criação de conjunto vazio.

No caso de um dos conjuntos ser inválido, o programa deverá informar o usuário qual conjunto é inválido e aguardar que o usuário pressione [ENTER] para continuar. E, pedir novamente que informe os dois conjuntos.

### Opção: 6 - Exibir listas.

Esta opção deverá imprimir o índice de todas as árvores $a_i$ e aguardar que o usuário pressione [ENTER] para voltar para o *menu* principal.

### Opção: 7 - Exibir todos os elementos de um conjunto.

Nesta opção, o programa deverá pedir ao usuário que informe o índice da árvore $a_i$ e em seguida deverá imprimir todos os elementos desta árvore, fazendo um percurso **pré-ordem**.

A cada 25 elementos, o programa deverá aguardar que o usuário pressionar [ENTER] para exibir os próximos 25 elementos do conjunto.

Ao final, após exibir todos os elementos, o programa deverá aguardar que o usuário pressionar [ENTER] para retornar ao *menu* principal.

No caso de um dos conjuntos ser inválido, o programa deverá informar que o conjunto é inválido e aguardar que o usuário pressione [ENTER] para continuar e retornar para o *menu* principal.

### Opção: 8 - Exibir altura da árvore/árvore balanceada.

Nesta opção, o programa deverá pedir ao usuário que informe o índice da árvore $a_i$ e em seguida deverá imprimir a altura da árvore e informar se está balanceada ou não.

No caso do conjunto ser inválido, o programa deverá informar que o conjunto é inválido e aguardar que o usuário pressione [ENTER] para continuar e retornar para o *menu* principal.

### Opção: F - Fim do programa.

Nesta opção, o programa deverá desalocar todos os dados alocados durante sua execução e, em seguida, deverá finalizar, retornando o console do S.O.