MINISTÉRIO DA EDUCAÇÃO
Universidade Federal de Alfenas . UNIFAL-MG
Rua Gabriel Monteiro da Silva, 700 . Alfenas/MG . CEP 37130-000
Fone: (35) 3299-1000 . Fax: (35) 3299-1063

Bacharelado em Ciência da Computação
Trabalho 1 – Estrutura de Dados I
Prof. Luiz Eduardo da Silva

Objetivo:
Utilizar a estrutura de dados LISTA ENCADEADA por alocação dinâmica de nós, na resolução de um problema de programação.

Procedimento:
Leia com atenção o enunciado do problema abaixo, desenvolva o programa C para resolver o problema descrito. Após a implementação, teste o programa para diferentes sequências de uso.

Problema:
A empresa de software XYZ tem um arquivo texto com informações dos seus funcionários e os projetos nos quais estão alocados, indicando inclusive o tempo em horas de alocação em cada projeto. Esse arquivo de entrada está organizado da seguinte forma:

```
#func
Nome1 #proj idProj1 tempoProj1 idProj2 tempoProj2 …
Nome2 #proj idProj1 tempoProj1 idProj2 tempoProj2 …
Nome3 #proj idProj1 tempoProj1 idProj2 tempoProj2 …
…
Nomen #proj idProj1 tempoProj1 idProj2 tempoProj2 …
```

Essa lista de associações pode ser representada numa estrutura encadeada de nós, conforme ilustrado na Figura 1. Por exemplo, para o arquivo de projetos:

```
3
Bruno 3 4 8 3 10 5 5
Daniel 1 5 25
Alan 2 3 15 1 5
```

Temos a seguinte estrutura de lista de associações:

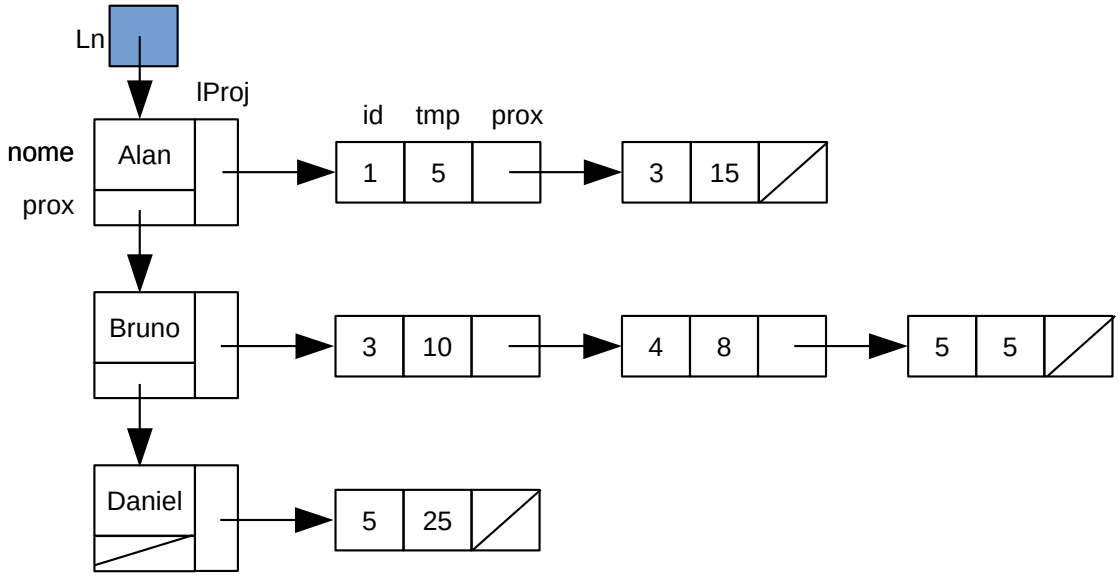

Figura 1: Representação de lista de associações

Observe que para facilitar a consulta, a lista de nomes de funcionários está ordenada e para cada funcionários estão associados os seus projetos (id=identificação e tmp=tempo). Nessa representação, temos duas listas encadeadas, a lista de nomes e a lista de projetos. As operações de inserção ordenada é a mesma nas duas listas, o que muda é como os nós estão representados.

Podemos pensar uma estrutura mais geral para esta lista encadeada, conforme representado na Figura 2, onde o campo de informação é um ponteiro genérico para um nó que pode ser tanto um (nome, listaProjetos) ou (idProj, tempo). O procedimento de inserção ordenada na lista é o mesmo, conforme ilustrado código em anexo.

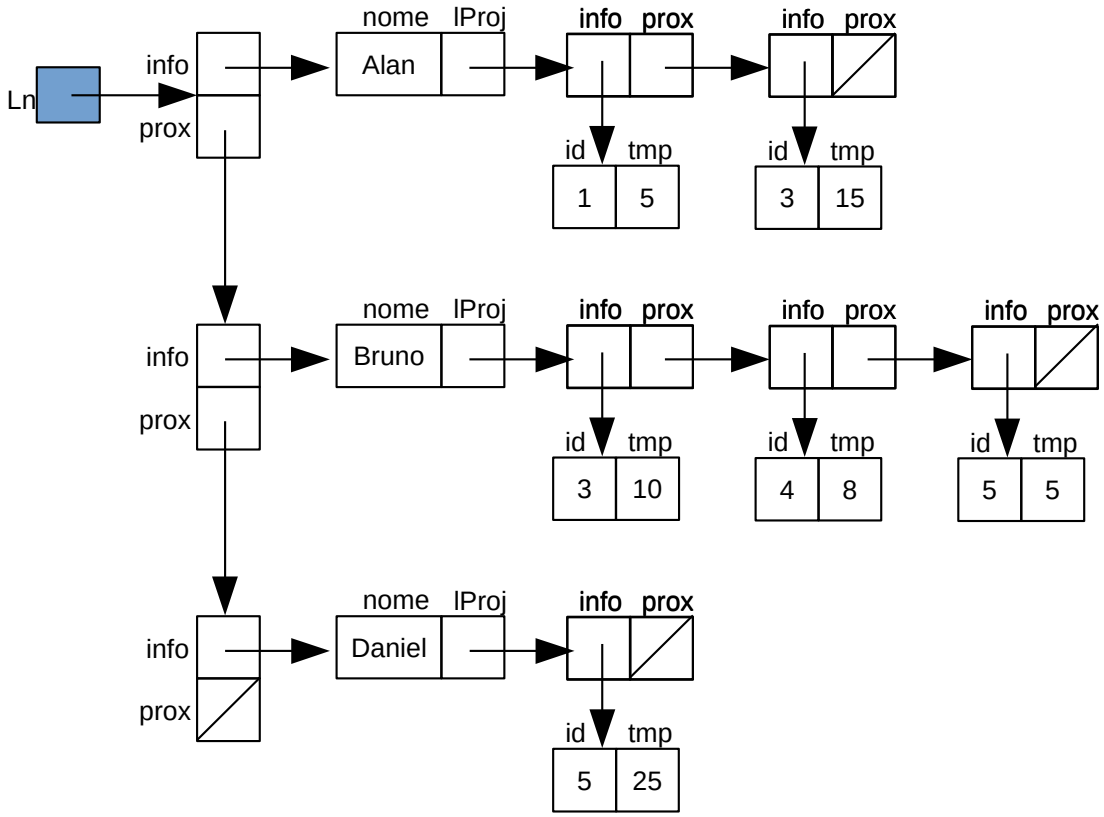

Figura 2: Lista genérica de Lista de Nome

Com essa lista construída a partir dos dados de entrada o seu trabalho é converter a lista para Projetos e Funcionários alocados, conforme ilustrado na Figura 3. E a partir dessas duas representações completar as funções que geram algumas estatísticas, como o projeto que está consumindo mais tempo, o funcionário que tem mais tempo alocado em projetos, o tempo total alocado em projetos e a percentagem do tempo alocado em projetos. Todas essas operações devem ser realizadas a partir do percurso nas representações de lista de nomes (Ln) e lista de projetos (Lp).

Roteiro:

1. Estudar o código já implementado para entender como a lista genérica está sendo implementada um linguagem C-Ansi.
2. Complete a implementação das funções:

```
PTno converte(PTno Ln);
void mostraPorProj(PTno Lp) ;
int projMaisTempo(PTno Lp);
void nomeMaisTempo(PTno Ln, char *nome);
int tempoTotal(PTno Ln);
void mostraPercAlocado(PTno Ln) ;
```

3. Teste todas as funções que foram implementadas e utilize alguns arquivos texto de teste para verificar o funcionamento do programa em questão. A saída deve ser conforme ilustrado nos exemplos apresentados.

Observação:

1. Incluir um comentário no cabeçalho de cada programa fonte com o seguinte formato:

```
/*--------------------------------------------------------------
 *          UNIFAL – Universidade Federal de Alfenas.
 * Trabalho..: Gerencia de projetos usando lista
 * Disciplina: Estrutura de Dados I
 * Professor.: Luiz Eduardo da Silva
 * Aluno.....: Fulano da Silva
 * Data......: DD/MM/AAAA
 *--------------------------------------------------------------*/
```

2. Enviar SOMENTE o programa principal (main.c) e/ou outros arquivos fonte que foram usados no projeto para a TAREFA de Envio de Arquivos no Moodle.

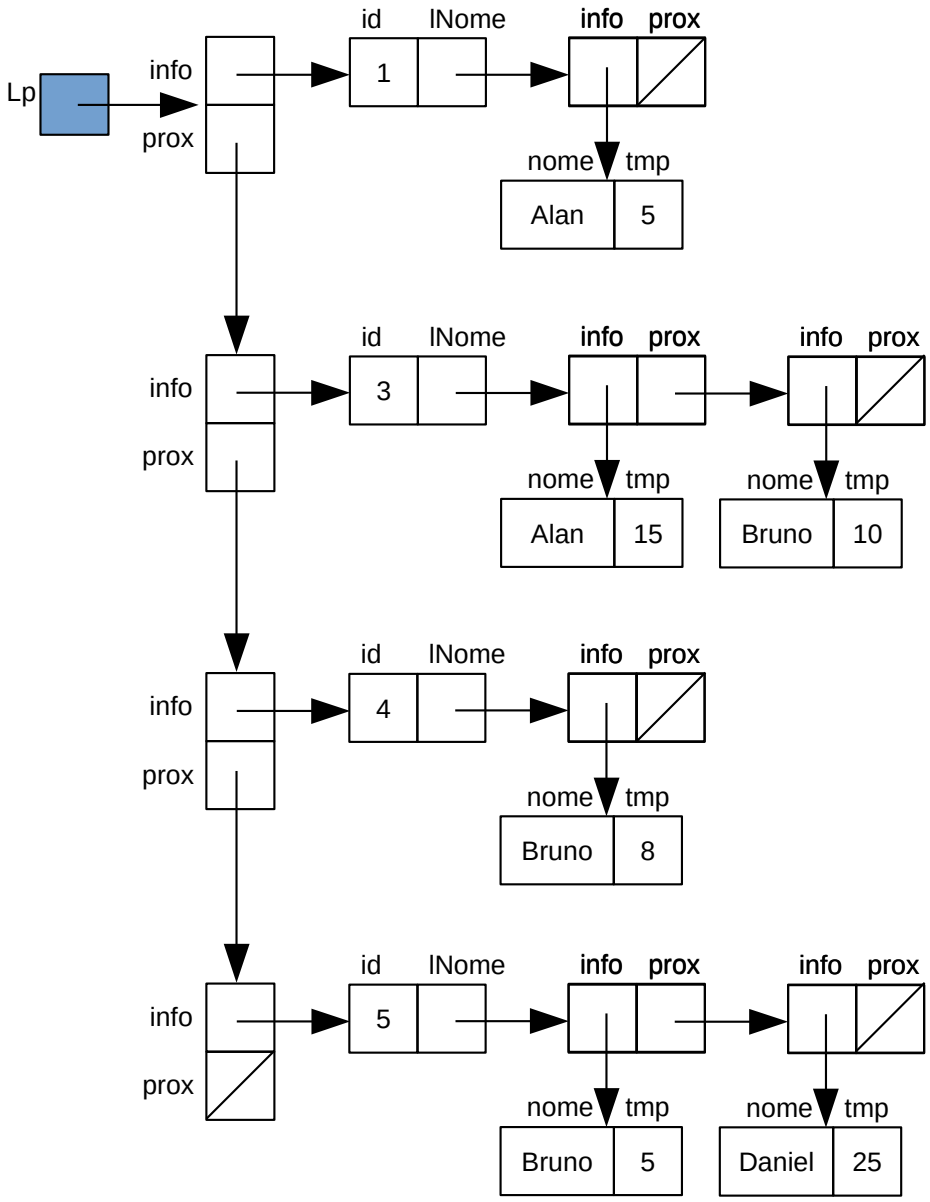

Figura 3: Lista de projetos

Anexo – Código inicial do trabalho

```
/*---------------------------------------------------------------
 *          UNIFAL – Universidade Federal de Alfenas.
 * Trabalho..: Gerencia de projetos usando lista
 * Disciplina: Estrutura de Dados I
 * Professor.: Luiz Eduardo da Silva
 * Aluno.....: Fulano da Silva
 * Data......: DD/MM/AAAA
 *---------------------------------------------------------------*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/*
 * ponteiros
 */
typedef struct nomeNo * PTnome; // pt estrutura nome
typedef struct projNo * PTproj; // pt estrutura projeto
typedef struct no * PTno; // pt no

/*
 * representacao da estrutura Nomes
 */
typedef struct nomeNo {
    char nome[10]; // nome do funcionario

    union {
        int tmp; // tempo OU
        PTno lProj; // lista de projetos
    };
} nomeNo;

/*
 * representacao da estrutura Projetos
 */
typedef struct projNo {
    int id; // identificacao do projeto

    union {
        int tmp; // tempo OU
        PTno lNome; // lista de nomes
    };
} projNo;

/*
 * representacao da lista encadeada
 */
typedef struct no {
    void * info;
    PTno prox;
} no;

/* Funcao geral de insercao numa lista encadeada ordenada geral
 * Os parâmetro são:
 * 1. L = a lista onde inserir
 * 2. N = o novo no a inserir
 * 3. cmp = funcao de comparacao (ID ou NOME)
 * A funcao retorna a lista modificada pela insercao
*/
PTno insere(PTno L, PTno N, int (*cmp)(const void *, const void *)) {
    PTno P = NULL, Q = L;
```

```
    while (Q && cmp(Q, N) < 0) {
        P = Q;
        Q = Q->prox;
    }
    if (P) {
        N->prox = P->prox;
        P->prox = N;
    } else {
        N->prox = L;
        L = N;
    }
    return L;
}

/* Funcao que retorna a comparacao de ID de nos
 * Retorna:
 *    negativo - se P->info->id MENOR QUE Q->info->id
 *    zero -     se P->info->id IGUAL     Q->info->id
 *    positivo - se P->info->id MAIOR QUE Q->info->id
 */
int compId(const void * P, const void * Q) {
    return ((PTproj) ((PTno) P)->info)->id -
             ((PTproj) ((PTno) Q)->info)->id;
}

/* Funcao que retorna a comparacao de NOMES de nos
 */
int compNome(const void * P, const void * Q) {
    return strcmp(((PTnome) ((PTno) P)->info)->nome,
            ((PTnome) ((PTno) Q)->info)->nome);
}

/* Funcao que carrega os nomes e os projetos onde esta alocado
 * Le o numero de pessoas.
 * Para cada pessoa:
 *    le o nome e o numero de projetos onde esta alocado
 *    Para cada projeto:
 *        le a identificacao e o tempo alocado no projeto

Exemplo de entrada de dados:
----------------------------
3
Bruno 3 4 8 3 10 5 5
Daniel 1 5 25
Alan 2 3 15 1 5

 */

PTno carregaDados(void) {
    int id, tmp, nNome, nProj;
    char nome[10];
    PTno lNome = NULL, lProj, aux;
    PTnome N;
    PTproj P;
    scanf("%d", &nNome);
    while (nNome) {
        scanf("%s", nome);
        scanf("%d", &nProj);
        lProj = NULL;
        while (nProj) {
            scanf("%d %d", &id, &tmp);
```

```
            // aloca estrutura projeto
            P = malloc(sizeof (projNo));
            P->id = id;
            P->tmp = tmp;
            // aloca no
            aux = malloc(sizeof (no));
            aux->info = P;
            lProj = insere(lProj, aux, compId);
            nProj--;
        }
        // aloca estrutura nome
        N = malloc(sizeof (nomeNo));
        strlcpy(N->nome, nome, 10);
        N->lProj = lProj;
        // aloca no
        aux = malloc(sizeof (no));
        aux->info = N;
        lNome = insere(lNome, aux, compNome);
        nNome--;
    }
    return lNome;
}

/* Funcao que mostra a lista de Pessoas e os projetos em
   que esta alocado
 Para os dados anteriores, apresenta:
 -----------------------------------
      Alan: [(1,5),(3,15)]
     Bruno: [(3,10),(4,8),(5,5)]
    Daniel: [(5,25)]
 */
void mostraPorNome(PTno Ln) {
    PTnome N;
    PTproj P;
    PTno Lp;
    while (Ln) {
        N = Ln->info;
        printf("%10s: [", N->nome);
        Lp = N->lProj;
        while (Lp) {
            P = Lp->info;
            printf("(%d,%d)", P->id, P->tmp);
            Lp = Lp->prox;
            if (Lp) printf(",");
        }
        printf("]\n");
        Ln = Ln->prox;
    }
}

/* Funcao que converte a Lista de Nomes por Projetos numa
 * lista de Projetos por Nomes. */
PTno converte(PTno Ln) {
    PTno Novo = NULL, Lp;
    // Acrescentar o codigo da funcao aqui
    return Novo;
}

/* Funcao que mostra a Lista de Projetos e as pessoas alocadas
 Para os dados anteriores, deve apresentar:
 ------------------------------------------
```

```
 1: [(Alan,5)]
 3: [(Alan,15),(Bruno,10)]
 4: [(Bruno,8)]
 5: [(Bruno,5),(Daniel,25)]
 */
void mostraPorProj(PTno Lp) {
    // Acrescentar o codigo da funcao aqui
}

/* Funcao que devolve o ID do projeto que consome mais tempo
 */
int projMaisTempo(PTno Lp) {
    int id, maior = 0;
    // Acrescentar o codigo da funcao aqui
    return id;
}

/* Funcao que retorna o nome do funcionario que tem mais tempo alocado
 */
void nomeMaisTempo(PTno Ln, char *nome) {
    int maior = 0;
    // Acrescentar o codigo da funcao aqui
}

/* Funcao que retorna o tempo total dos projetos
 */
int tempoTotal(PTno Ln) {
    int total = 0;
    // Acrescentar o codigo da funcao aqui
    return total;
}

/* Funcao que mostra o percentual de tempo alocado em projetos
 * Para os dados anteriores, deve apresentar:
      Alan - 29.41%
     Bruno - 33.82%
    Daniel - 36.76%
 */
void mostraPercAlocado(PTno Ln) {
    // Acrescentar o codigo da funcao aqui
}

int main(int argc, char** argv) {
    PTno listaNome = carregaDados();
    PTno listaProj = converte(listaNome);
    char nome[10];
    mostraPorNome(listaNome);
    mostraPorProj(listaProj);
    printf("Projeto mais longo = %d\n", projMaisTempo(listaProj));
    nomeMaisTempo(listaNome, nome);
    printf("Nome com mais projetos = %s\n", nome);
    printf("Tempo Total dos projetos = %d\n", tempoTotal(listaNome));
    mostraPercAlocado(listaNome);
    return (EXIT_SUCCESS);
}
```