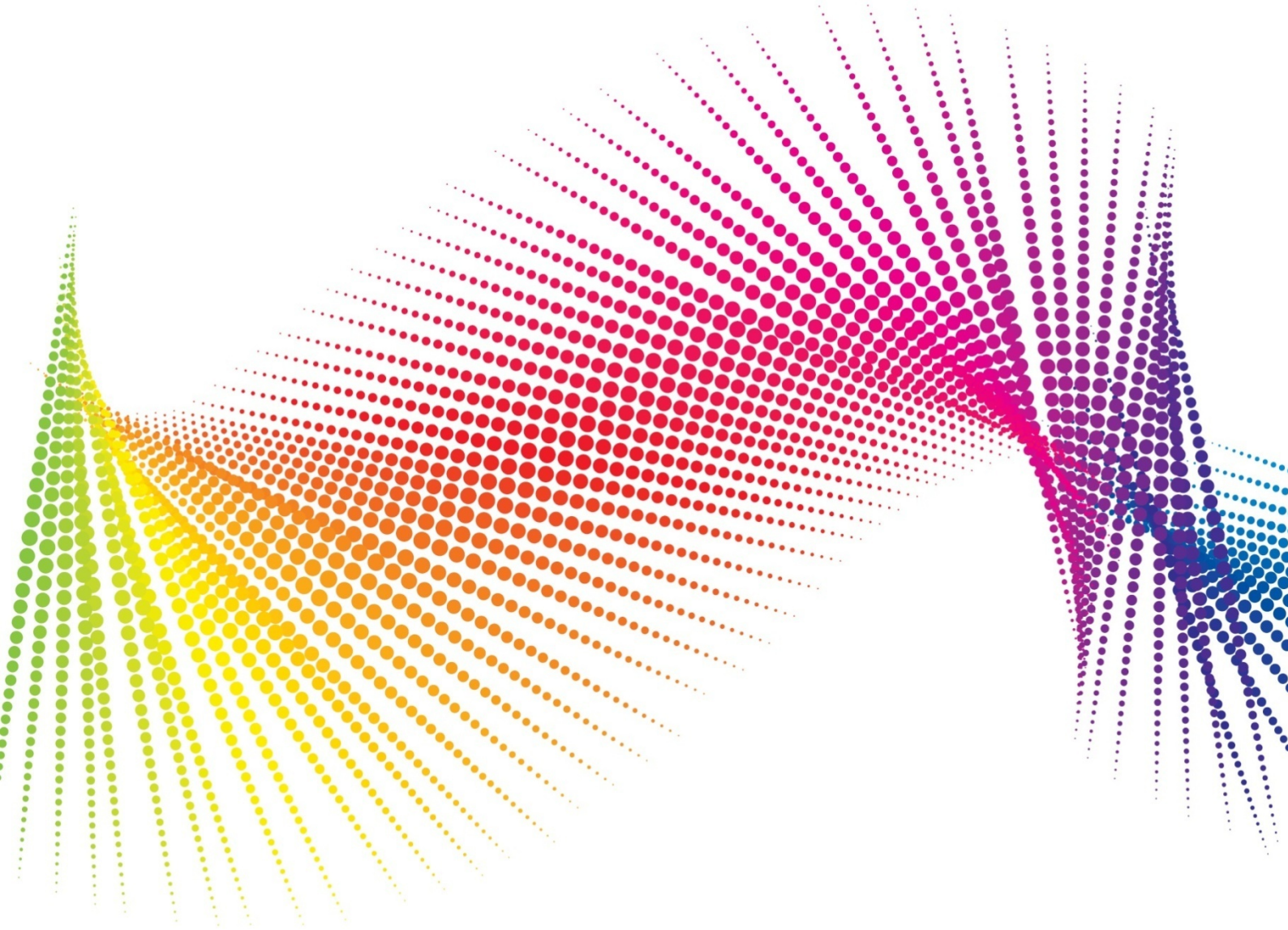
UNINOVE
Universidade Nove de Julho

Estrutura de Dados

Aula 20



Este material é parte integrante da disciplina oferecida pela UNINOVE.
O acesso às atividades, conteúdos multimídia e interativo, encontros virtuais, fóruns de discussão e a comunicação com o professor devem ser feitos diretamente no ambiente virtual de aprendizagem UNINOVE.

Uso consciente do papel.
Cause boa impressão, imprima menos.

Aula 20: Listas simplesmente encadeadas com alocação dinâmica

Objetivo: Continuar estudando as listas encadeadas, mostrando como implementar um llse com alocação dinâmica.

Lista linear simplesmente encadeada

Caro aluno, na aula 19 chegamos à conclusão que a solução apresentada era muito ineficiente, pois os nós ficaram dependentes do primeiro apontado por `inicio`, e para aumentarmos a lista, teríamos que fazer:

```
inicio->prox->prox->prox->prox .... = aux;
```

Aqui vamos apresentar uma solução completa que utiliza funções para INSERÇÃO, REMOÇÃO, LOCALIZAÇÃO e IMPRESSÃO de nós de uma lista.

Algoritmo para manipulação de uma lista linear simplesmente encadeada com alocação dinâmica – LLSE

```
/* llse.c:  Este programa cria uma llse-lista linear simplesmente encadeada
            com alocação dinâmica, ordenada por ordem crescente
*/

#include <stdio.h>
#include <conio.h>
#include <string.h>
#define N 10

// define o tipo no e os apontadores para manipulação da llse

typedef struct nodo { int info;
                      struct nodo *prox;
                    } no;
no *inicio, *novo, *ant, *aux;

int opcao, infosai;

// protótipos das funções

int menu (void);
```

```
void insereOrdenado (void);
void localiza (int novainfo);
void retira (int infosai);
void imprime (void);
int main ( )
{   system("cls");
    system("color 1b");

    printf("       Este programa le dados e monta uma estrutura de");
    printf("\n lista linear simplesmente encadeada com alocacao dinamica\n\n\n");
    printf("          <ENTER> para comecar ... ");
    getch();

    // inicializa o ponteiro início com "nulo"
    inicio = NULL;
   do {
        opcao = menu();
        switch (opcao)
        {
          case 1:
                    system("cls");
                    printf("OPCAO 1- Insere ordenado na lista simplesmente encadeada \n\n");

                    insereOrdenado();
                    imprime();
                    printf("\nPressione ENTER para continuar...");
                    getch();
                    break;

            case 2:
                    system("cls");
                    printf(" OPCAO 2 - Retira da lista simplesmente encadeada \n\n");
                    printf("\nElemento a ser retirado da lista: ");
                    scanf("%d",&infosai);
                    retira(infosai);

                    imprime();
                    printf("\nPressione ENTER para continuar...");
                    getch();
                    break;

            case 3:
                    system("cls");
                    printf(" OPCAO 3 - Imprime lista simplesmente encadeada \n\n");

                    imprime();
```

```
                printf("\nPressione ENTER para continuar...");
                getch();
                break;

        case 0:
                printf("\nFim do programa. Pressione uma tecla ...");
                getch();
                break;

        default: printf("\n Opcao invalida... ");
                getch();

        } // switch

       } while (opcao!=0);

        // dealoca todos os ponteiros usados no programa
       free(inicio);
       free(novo);
       free(aux);
       free(ant);
       return 0;

}
int menu (void)
{       int opcao;

                //apresenta as opções para manipulação da llse
                system("cls");

                printf("Voce deseja: ");
                printf("\n 1- Inserir ordenado na lista simplesmente encadeada");
                printf("\n 2- Retirar um no da lista simplesmente encadeada");
                printf("\n 3- Listar os nos da lista simplesmente encadeada");
                printf("\n 0- Finalizar ");
                printf("\n Opcao: ");
                scanf("%i",&opcao);
                return(opcao);
}

void insereOrdenado ()
{       //aloca espaço para o novo nó e solicita a informação que entrará na llse
        novo = malloc (sizeof(no));
        printf("\nDigite o valor (inteiro) a ser inserido na lista: " );
        scanf("%d", &novo-> info );
        novo->prox = NULL;
        if (inicio == NULL)
```

```
            // primeiro nó da lista basta inicio apontar para ele
            inicio = novo;
        else
        {   // não é o primeiro nó, então deve-se localizar o nó
            // ou o lugar onde ele deve ser inserido
            localiza (novo-> info);

            if (ant == NULL)
                // o nó é o menor de todos, então entra no início da lista
                inicio = novo;
            else
                // nó entra entre dois nós ou no final da lista
                ant->prox=novo;

            // finaliza o encadeamento do novo nó
            novo-> prox = aux;
        }
}
void localiza (int novainfo)
{       int achou;
        // localiza() tenta achar o nó com a infromação novainfo a partir do inicio da lista

        ant = NULL;
        aux = inicio;
        achou=0;
        while (aux != NULL && achou==0)
        {       if (aux->info >= novainfo)
                        achou = 1;
                else  {
                        ant = aux;
                        aux = aux->prox;
                      }
   }
}
void retira(int infosai)
{   // antes de retirar um nó, chama-se localiza para tentar localizá-lo na lista
    localiza (infosai);

    if (aux == NULL)
    {
        printf ("\n Lista vazia ou informacao nao encontrada na lista \n");
        getch();
    }
    else if (aux->info != infosai)
        {       printf ("\n Informacao nao pertence a lista \n");
                getch();
        }
```

- ao finalizar a localização, aux aponta para o nó com a informação novainfo
- ou para o lugar onde esta informação deva entrar na lista
-ant aponta para o nó anterior a aux
-se ant finaliza com NULL, significa que aux parou no primeiro nó da lista

Se localiza retorna aux igual a NULL, indica que o nó com a informação. infosai não está na lista ou a lista está vazia

Se a informação apontada por aux não for igual a infosai. Significa que infosai não está na llse

```
        else  {             if (ant != NULL)
                        // retira-se o nó aux, que está entre 2 nós ou é o último nó
                        ant->prox = aux->prox;
                    else
                        // retira-se o nó aux, que é o primeiro nó da lista
                        inicio = aux->prox;
                    free (aux);
                    printf("\nSucesso na retirada!");
                    getch();
                  }
}

void imprime()
{       aux = inicio;
        printf ("\n Lista Linear Simplesmente Encadeada: \n");
        if (inicio == NULL)
                printf("\n A lista esta vazia! \n");
        else
        {   while( aux != NULL )
            {       printf("%i -> ", aux-> info );
                    aux = aux->prox;
            }
           printf("-// \n\n");
           getch();
        }
}
```

Caro aluno, agora que você entendeu bem este programa que mostra como criar e manipular uma lista linear simplesmente encadeada com alocação dinâmica, implemente e teste este programa. Acesse o AVA e assista à animação que mostra os ponteiros **inicio**, **aux**, **novo** e **ant** sendo usados na construção de uma llse pelo programa apresentado.

Agora, caro aluno, vamos praticar resolvendo os exercícios propostos. **Leia a lista, resolva os exercícios e verifique seu conhecimento**. Caso fique alguma dúvida, leve a questão ao Fórum e divida com seus colegas e professor.

REFERÊNCIAS

SCHILDT, H. *C completo e total*. São Paulo: Makron Books, 1997.

TENEMBAUM, Aaron M., et al. *Estruturas de dados usando C.* São Paulo: Pearson Makron Books, 1995.