Linguagem C: Listas Encadeadas

Prof. Paulo R. S. L. Coelho
paulo@facom.ufu.br

Faculdade de Computação
Universidade Federal de Uberlândia

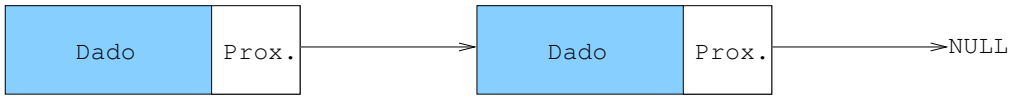
GEQ007

Organização

Organização

Introdução

- Uma lista encadeada (ou lista ligada) é uma representação de uma **sequência** de objetos na memória do computador.
- Cada elemento é armazenada em uma **célula** ou **nó** da lista.
- De maneira simplificada, um nó é composto de duas partes:
  - a informação (ou o dado) de interesse; e
  - uma referência para o próximo nó.



Vantagens e Desvantagens

- O principal benefício de uma lista encadeadas em relação a vetores é o fato de que os elementos de uma lista podem ser facilmente inseridos ou removidos.
- E isso pode ser feito sem necessidade de realocação ou reorganização de toda a estrutura, uma vez que os nós não precisam ser armazenados em sequência na memória.
- Outro ponto importante é a facilidade de inserção e remoção de nós em qualquer ponto da lista, tomados os devidos cuidados nas atualizações das referências.
- Por outro lado, listas encadeadas por si só não permite acesso direto a um dado, ou qualquer forma eficiente de indexação. Assim, muitas operações básicas, como buscar um nó com uma determinada informação, podem significar percorrer a maioria ou todos os elementos da lista.

Organização

Listas Encadeadas em C I

- Listas encadeadas são representadas em C utilizando-se estruturas (`struct`).
- A estrutura de cada célula de uma lista ligada pode ser definida da seguinte maneira:

```
struct cel {
  int dado;
  struct cel *prox;
};
```

Uma outra maneira de representar, utilizando `typedef`, seria:

```
typedef struct cel celula;
struct cel {
  int dado;
  celula *prox;
};
```

Listas Encadeadas em C II

- Uma célula `c` e um ponteiro `p` para uma célula podem ser declarados assim:

```
celula  c;
celula *p;
```

- Se `c` é uma célula, então `c.dado` é o conteúdo da célula e `c.prox` é o endereço da próxima célula.
- Se `p` é o endereço de uma célula, então `p->dado` é o conteúdo da célula e `p->prox` é o endereço da próxima célula.
- Se `p` é o endereço da última célula da lista, então `p->prox` vale `NULL`.
- O endereço de uma lista encadeada é o endereço de sua primeira célula. Se `p` é o endereço de uma lista, pode-se dizer simplesmente ”p é uma lista”.

Inserção

- Todas as operações serão apresentadas considerando-se que a lista possui um nó inicial, cujo valor não se tem interesse, denominado ”cabeça”da lista. Esse nó tem apenas a função de apontar para o primeiro elemento inserido na lista.
- A função a seguir deve inserir uma nova célula com conteúdo `x` após a posição apontada por `p` (`p` não pode ser nulo).

```
void insere (int x, celula *p)
{
   celula *nova;
   nova = (celula *) malloc (sizeof(celula));
   nova->dado = x;
   nova->prox = p->prox;
   p-> = nova;
}
```

Impressão

- A função seguinte imprime uma lista a partir da posição apontada por `ini->prox`.

```
void imprime(celula *ini)
{
   celula *p;
   for (p = ini->prox; p != NULL; p = p->prox) {
      printf ("%d\t", p->dado);
   }
   printf ("\n");
}
```

Organização

Exercício I

1. Implemente um programa em C que utiliza a estrutura apresentada para implementar uma lista. O programa deve mostrar ao usuário duas opções. Se o usuário escolher 1, a lista deve ser impressa; se escolher 2, ele deve entrar com o valor do conteúdo do novo elemento da lista.

Resposta I

```
#include<stdio.h>
#include<stdlib.h>

typedef struct cel celula;
struct cel {
  int dado;
  celula *prox;
};

void insere (int x, celula *p)
{
   celula *nova;
   nova = (celula *) malloc (sizeof(celula));
   nova->dado = x;
   nova->prox = p->prox;
   p->prox = nova;
}

void imprime(celula *ini)
{
   celula *p;
   printf("\nValores na lista:\n");
```

Resposta II

```
    for (p = ini->prox; p != NULL; p = p->prox) {
        printf ("%d\t", p->dado);
    }
    printf ("\n");
}

int main() {
    int op = -1, valor;
    celula *lista = NULL;
    lista = (celula *) malloc(sizeof(celula));

    while (op != 0) {
        printf("\nOpções disponíveis:\n");
        printf("\t1 p/ imprimir lista.\n");
        printf("\t2 p/ inserir novo elemento na lista.\n");
        printf("\t0 p/ encerrar.\n");
        printf("Entre opção desejada:  ");
        scanf("%d", &op);
        switch(op) {
            case 0:
                printf("\n\nTCHAU!\n");
                break;
            case 1:
```

Resposta III

```
                imprime(lista);
                break;
            case 2:
                printf("\nEntre valor a ser inserido na lista: ");
                scanf("%d", &valor);
                insere(valor, lista);
                break;
            default:
                printf("\n\nOPCAO INVALIDA!\n");
        }
    }
    return 0;
}
```