Estrutura de Dados

Profa. Juliana Kaizer Vizzotto

Aula 3

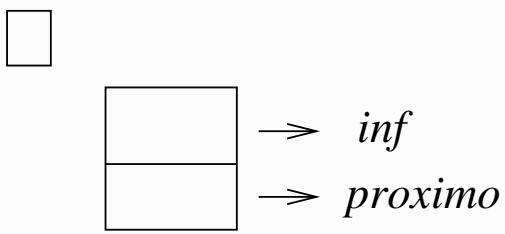
Listas Ligadas

Motivação

- Matrizes?? Limitações??
- Seu tamanho tem que ser conhecido no momento da compilação.
- Os dados em uma matriz estão separados na memória pela mesma distância.

Solução: estruturas ligadas!

1. Coleção de nós, que armazenam dados.
2. Ligações com outros nós.



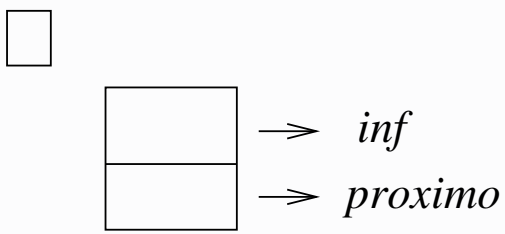
Listas Ligadas

Motivação

- Matrizes?? Limitações??
- Seu tamanho tem que ser conhecido no momento da compilação.
- Os dados em uma matriz estão separados na memória pela mesma distância.

Solução: estruturas ligadas!

1. Coleção de nós, que armazenam dados.
2. Ligações com outros nós.



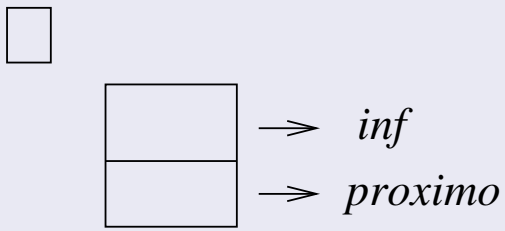
Listas Ligadas

Motivação

- Matrizes?? Limitações??
- Seu tamanho tem que ser conhecido no momento da compilação.
- Os dados em uma matriz estão separados na memória pela mesma distância.

Solução: estruturas ligadas!

1. Coleção de nós, que armazenam dados.
2. Ligações com outros nós.



Listas Singularmente Ligadas

Um nó tem um vínculo somente para o seu sucessor.

- Como acessamos qualquer nó na lista?
- Uma variável *p* é utilizada para acessar qualquer elemento da lista!
- Um nó contém dois membros de dados:
  1. *info*: que é utilizado para estocar informação e é importante para o usuário; e
  2. *proximo*: que é utilizado para vincular nós que formam uma lista ligada.

Listas Singularmente Ligadas

Um nó tem um vínculo somente para o seu sucessor.

- Como acessamos qualquer nó na lista?
- Uma variável *p* é utilizada para acessar qualquer elemento da lista!
- Um nó contém dois membros de dados:
  1. *info*: que é utilizado para estocar informação e é importante para o usuário; e
  2. *proximo*: que é utilizado para vincular nós que formam uma lista ligada.

Listas Singularmente Ligadas

Implementando nodos

```
struct lista{
    int info;
    struct lista* prox;
};
typedef struct lista Lista;
```

- Note que trata-se de uma estrutura auto-referenciada, pois, além do campo que armazena a informação, há um campo que é um ponteiro para uma estrutura do mesmo tipo.
- O tipo `Lista` representa um nó da lista e a estrutura de lista encadeada é representada por um ponteiro para o seu primeiro elemento.

Listas Singularmente Ligadas

Função de Inicialização

```
/* funcao de inicializacao */
Lista* inicializa(void){
 return NULL;
}
```

- A função que inicializa uma lista deve criar uma lista vazia, sem nenhum elemento. Como uma lista é representada por um ponteiro para o primeiro elemento, a lista vazia é representada pelo ponteiro `NULL`.

Listas Singularmente Ligadas

Função de Inserção

```
/*Insere no incio */
Lista* insere(Lista* l, int i){
   Lista* novo =(Lista*) malloc(sizeof(Lista));
   novo -> info =i;
   novo -> prox = l;
   return novo;
}
```

Uma vez criada a lista, podemos inserir novos elementos nela. Para cada elemento devemos alocar dinamicamente memória para o mesmo e encadeá-lo na lista. A função de inserção mais simples insere o elemento no início da lista.

Listas Singularmente Ligadas

Exemplo

- Para ilustrar vamos criar uma lista inicialmente vazia e inserir novos elementos nela.

Listas Singularmente Ligadas

Exemplo:

```
 int main(void){
  Lista* l;
  l = inicializa();
  l = insere(l,30);
  l = insere(l,18);
  ...
  return 0;
}
```

Observe que não podemos deixar de atualizar a variável que representa a lista a cada nova inserção.

Listas Singularmente Ligadas

Função que percorre os elementos da lista

```
void imprime(Lista* l){

   Lista* p;
   for(p=l;p!=NULL;p=p->prox)
     printf("Info = %d\n", p->info);
}
```

Listas Singularmente Ligadas

Função que verifica se a lista está vazia

```
int vazia(Lista* l){
  if (l==NULL)
    return 1;
  else
    return 0;
}
```

Listas Singularmente Ligadas

Função de busca

```
Lista* busca(Lista* l, int v){
  Lista* p;
  for(p=l;p!=NULL;p=p->prox)
   if (p->info == v)
     return p;
  return NULL;
}
```

Busca um elemento na lista. Retorna um ponteiro para o elemento, caso ele tenha sido encontrado.

Listas Singularmente Ligadas

Função que Retira um elemento da lista

```
Lista* retirna(Lista* l, int v){
  Lista* ant = NULL;
  Lista* p = l; // ponteiro para percorrer a lista
  while(p !=NULL && p-> info != v){
    ant = p;
    p = p ->prox;}
  if (p == NULL)
    return l; //nao achou
  if (ant == NULL){
    //retira do inicio
    l = p-prox;
  } else { //retira do meio
    ant-prox = p-> prox;
  }  return l;}
```

Listas Singularmente Ligadas

Função para liberar a lista

```
void libera(Lista* l){
  Lista* p = l;
  while(p != NULL){
    Lista* t = p->prox;
    free(p);
    p = t;
  }
}
```

Listas Singularmente Ligadas

Exemplo de uso das funções

```
#include<stdio.h>
int main(){
  Lista* l;
  l = inicializa();
  l = insere(l,23);
  l = insere(l,45);
  l = insere(l,56);
  l = insere(l,78);
  imprime(l);
  l = (retira(l,78);
  imprime(l);
  l = retira(l,45);
  imprime(l);
  libera(l);
  return 0;
```