Programação I

Listas duplamente ligadas

Estrela Ferreira Cruz

1

Objetivos da aula

Objetivos da aula:

Listas duplamente ligadas:

- Apresentação do conceito;
- Definição da estrutura de dados que representa uma lista duplamente ligada.
- Vantagens de desvantagens do uso de listas duplamente ligadas em relação a outras estruturas de dados.
- Operações de manuseamento: inserção (início, meio e fim), remoção (início, meio e fim), percorrer a lista, etc.
- Apresentação de exemplos práticos.

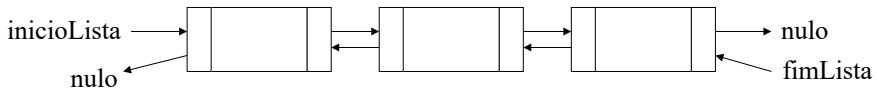
2

2

Listas duplamente ligadas

Listas duplamente ligadas

- Entre as listas ligadas, existem listas com ligações múltiplas, e dentro destas, um caso especial que é o da lista em que cada elemento tem duas ligações que apontam em direções opostas: as listas duplamente ligadas.
- Uma lista duplamente ligada é uma lista em que cada registo tem um campo que aponta para o elemento seguinte na lista e outro que aponta para o elemento anterior.
- ou seja, cada elemento de uma lista duplamente ligada contém o endereço de memória do próximo elemento da lista e o endereço de memória o elemento anterior na lista (além da informação "útil").



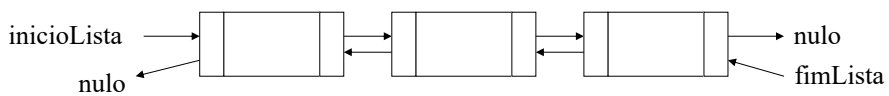
3

3

Listas duplamente ligadas

Listas duplamente ligadas:

- Uma das vantagens das listas duplamente ligadas, é o facto de nos podermos deslocar para o nó anterior ou para o nó seguinte, a um determinado nó com igual facilidade.
- Para além disso, uma lista duplamente ligada permite-nos remover um nó qualquer da lista em tempo constante, usando apenas um apontador para esse nó.
- No entanto, uma lista duplamente ligada duplica o espaço necessário para guardar ligações (apontadores para os nós vizinhos) e duplica o número de manipulações de ligações por cada operação básica.



4

4

Listas duplamente ligadas

- As listas duplamente ligadas podem ser percorridas nos dois sentidos.
- Para isso é necessário reservar um apontador para o inicio da lista e outro para o fim da lista.
- Assim sendo, a estrutura de dados de uma lista de alunos, pode ser a seguinte:

```
typedef struct info{
        int numero;
        char nome[100];
 } INFO;
typedef struct Elem {
        INFO node;
        struct Elem *proximo;
        struct Elem *anterior;
}ELEMENTO;
int main() {
ELEMENTO *iniLista=NULL, *fimLista=NULL;
……
}
```

5

5

Listas duplamente ligadas

Operações sobre listas duplamente ligadas:

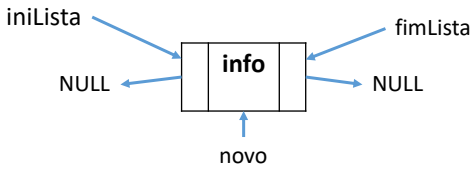
- A inserção no inicio da lista
- Inserção no fim da lista
- Inserção no meio da lista
- Imprimir para o ecrã o conteúdo da lista.
- Remoção de um elemento da lista (inicio, meio e fim)
- Devolver o elemento anterior a um determinado elemento
- Calcular o tamanho da lista
- Etc.

6

6

Listas ligadas

Inserir o primeiro elemento na lista

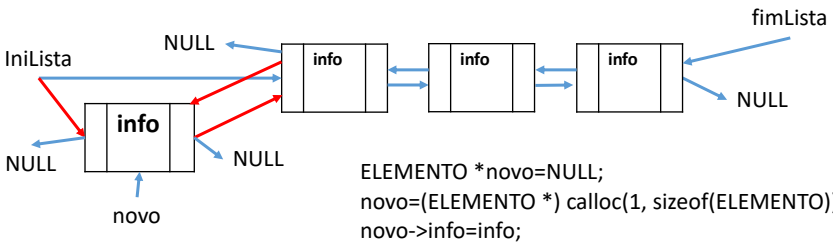



```
ELEMENTO *novo=NULL;
novo=(ELEMENTO *) calloc(1, sizeof(ELEMENTO));
novo->info=info;
novo->proximo=NULL;
novo->anterior=NULL;
iniLista=novo;
fimLista=novo;
```


7

Listas duplamente ligadas

Inserir um elemento no **início da lista** que já contem outros elementos




```
ELEMENTO *novo=NULL;
novo=(ELEMENTO *) calloc(1, sizeof(ELEMENTO));
novo->info=info;
novo->proximo=NULL;
novo->anterior=NULL;
novo->proximo=iniLista;
Inilista->anterior=novo;
iniLista=novo;
```


8

Listas d

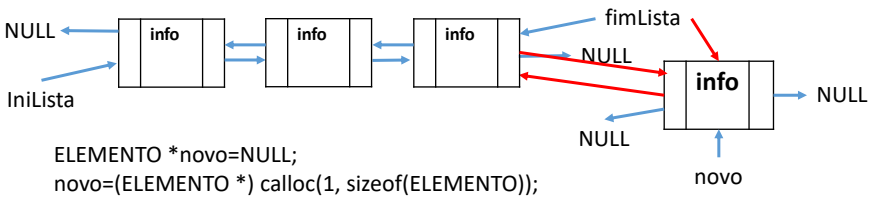
A inserção no início ou no fim da lista são muito idênticas.

```
int InserirInicioLista (ELEMENTO **iniLista,
ELEMENTO **fimLista,INFO newInfo){
  ELEMENTO *novo=NULL;
  novo=(ELEMENTO *)malloc(sizeof(ELEMENTO));
  if (novo==NULL) {
      printf("Out of memory\n"); return -1;
  }
  novo->node = newInfo;
  novo->anterior= NULL;
  novo->proximo= NULL;
  if (*iniLista==NULL) {
      *iniLista=novo;
      *fimLista=novo;
  }
  else {
      novo->proximo = *iniLista;
      (*iniLista)->anterior=novo;
      *iniLista=novo;
  }
  return 0;
}
```

9

Listas duplamente ligadas

Inserir um elemento no **fim da lista** que já contem outros elementos




```
ELEMENTO *novo=NULL;
novo=(ELEMENTO *) calloc(1, sizeof(ELEMENTO));
novo->info=info;
novo->proximo=NULL;
novo->anterior=NULL;
novo->anterior=fimLista;
fimLista->proximo=novo;
fimLista=novo;
```


10

Listas d

A inserção no fim da lista.

```
int InserirFimLista (ELEMENTO **iniLista,
ELEMENTO **fimLista, INFO newInfo){
  ELEMENTO *novo=NULL;
  novo=(ELEMENTO *)malloc(sizeof(ELEMENTO));
  if (novo==NULL) {
      printf("Out of memory\n"); return -1;
  }
  novo->node = newInfo;
  novo->anterior= NULL;
  novo->proximo= NULL;
  if (*fimLista==NULL) {
      *iniLista=novo;
      *fimLista=novo;
  }
  else {
      novo->anterior=*fimLista;
      (*fimLista)->proximo=novo;
      *fimLista=novo;
  }
  return 0;
}
```

11

Listas duplamente ligadas

Para listar o conteúdo de uma lista para o ecrã, podemos começar pelo inicio ou pelo fim. Para começar pelo inicio (duas opções):

```
void ListarSeg (ELEMENTO *iniLista){
ELEMENTO *aux=iniLista;
If (iniLista==NULL){printf("Lista vazia:\n"); return;}
while (aux!=NULL) {
     printf("%d, %s\n", aux->node.num, aux->node.nome);
    aux = aux->proximo;
}
}
```

```
void ListarSeg2 (ELEMENTO *iniLista){
ELEMENTO *aux=iniLista;
If (iniLista==NULL){printf("Lista vazia:\n"); return;}
for(aux=iniLista; aux!=NULL; aux = aux->proximo) {
     printf("%d, %s\n", aux->node.num, aux->node.nome);
}
}
```

12

12

Listas duplamente ligadas

Para listar o conteúdo de uma lista para o ecrã, podemos começar pelo inicio ou pelo fim. Para começar pelo fim (duas opções):

```
void ListarAnt (ELEMENTO *fimLista){
ELEMENTO *aux=fimLista;
if (fimLista==NULL){printf("Lista vazia:\n"); return;}
while (aux!=NULL) {
     printf("%d, %s\n", aux->node.num, aux->node.nome);
     aux = aux->anterior;
}
}
```

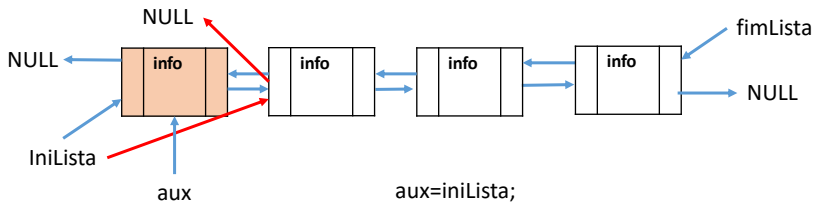
```
void ListarAnt2 (ELEMENTO *fimLista){
ELEMENTO *aux=fimLista;
if (fimLista==NULL){printf("Lista vazia:\n"); return;}
for(aux=fimLista; aux!=NULL; aux = aux->anterior) {
     printf("%d, %s\n", aux->node.num, aux->node.nome);
}
}
```

13

13

Listas duplamente ligadas

Remover o primeiro elemento da lista.



aux=iniLista;

aux->proximo->anterior=NULL;

iniLista=aux->proximo;

free(aux);

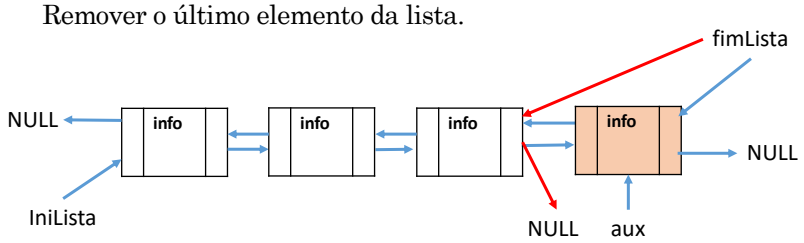
EstrelaFCruz

14

14

Listas duplamente ligadas

Remover o último elemento da lista.

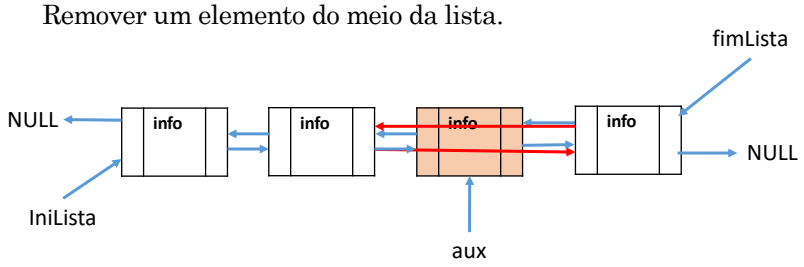



```
aux=fimLista;
aux->anterior->proximo=NULL;
fimLista=aux->anterior;
free(aux);
```


15

Listas duplamente ligadas

Remover um elemento do meio da lista.




```
aux->anterior->proximo=aux->proximo;
aux->proximo->anterior=aux->anterior;
free(aux);
```


16

Listas duplamente ligadas

Para remover um elemento da lista, não é necessário guardar o nó anterior numa variável auxiliar.

```
Int removeElem(ELEMENTO **iniLista, ELEMENTO **fimLista, int num) {
ELEMENTO *aux=*iniLista;

while(aux!=NULL && aux->node.numero != num) {
          aux = aux ->proximo;
}
if (aux ==NULL) return -1;       // não existe elemento num ou a lista é vazia
if (aux->anterior == NULL) {     // vai remover o 1º elemento
          *iniLista= aux ->proximo;
          if (*iniLista!= NULL) {(*iniLista)->anterior=NULL;}
}
else {aux->anterior->proximo = aux->proximo; }

if (aux->proximo == NULL) {    // vai remover o ultimo elemento
          *fimLista= aux ->anterior;
          if (*fimLista!=NULL) {(*fimLista)->proximo=NULL;}
}
else {aux->proximo->anterior = aux->anterior;}
free(aux); return 0;
}
```

17

Listas duplamente ligadas

Devolver o elemento anterior ao elemento numAct

```
ELEMENTO *anterior (ELEMENTO *iniLista,int numAct) {
 ELEMENTO *aux=iniLista;
 while(aux!=NULL && aux->node.numero != numAct){
       aux = aux->proximo;
 }
 if (aux==NULL) return NULL; //não encontrou o numAct
 return(aux->anterior);
}
```

18

18

Listas duplamente ligadas

Função que permite libertar memória ocupada pelos elementos da lista.

```
void libertaMemlista(Elemento **iniLista, Elemento
**fimLista){
        Elemento *aux=*iniLista, *proximo=NULL;
        *iniLista=NULL;
        *fimLista=NULL;
        while(aux!=NULL) {
                proximo = aux->proximo;
                free(aux);
                aux=proximo;
        }
}
```

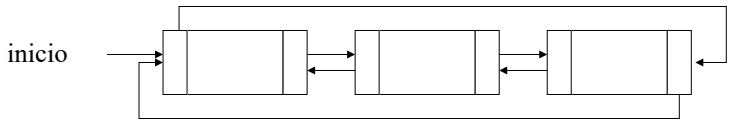
19

19

Listas duplamente ligadas circulares

Listas duplamente ligadas circulares

Nestas listas, o apontador para o próximo elemento da lista, do último elemento, não está apontar para NULL, mas está apontar para o inicio da lista. Da mesma forma o apontador para o anterior do inicio da lista também não está apontar para NULL, mas está apontar para o último elemento da lista.



20

20

Listas Ligadas

Exercício:

1 – Implemente uma **lista duplamente ligada** que permita fazer a gestão (inserir/remover/listar e alterar) das inscrição de alunos da ESTG no workshop sobre Data Science. Sobre cada inscrição devemos armazenar o **nome**, **numero, curso** e **ano que frequenta** do aluno.

Bibliografia

- Programação Avançada Usando C, António Manuel Adrego da Rocha, ISBN: 978-978-722-546-0.
- Schildt, Herbert: C the complete Reference, McGraw-Hill, 1998.
- Algoritmia e Estruturas de Dados, José Braga de Vasconcelos, João Vidal de Carvalho, ISBN: 989-615-012-5.
- Elementos de Programação com C - Pedro João Valente D. Guerreiro, 3ª edição, ISBN: 972-722-510-1.
- Introdução à Programação Usando C, António Manuel Adrego da Rocha, ISBN: 972-722-524-1.