estgoh
Politécnico de Coimbra

Estruturas de Dados | Programação II

Engenharia Informática | Sistemas e Tecnologias de Informação

1º ano, 2º semestre

2022–2023

estgoh
Politécnico de Coimbra

Listas ligadas ou Listas dinâmicas

As listas ligadas são um tipo de estrutura de dados mais versátil com **dimensão ajustável**.

Cada elemento da lista (nó) é, habitualmente, uma estrutura composta por dois campos:

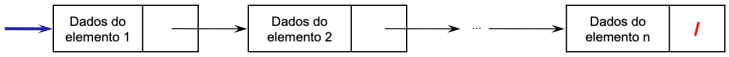
- informação (inteiro, real, tabela, estrutura composta,...)
- ponteiro para o elemento seguinte

O ponteiro do último elemento da lista deve ser NULL (/) para indicar o fim da lista.

Não é necessário guardar o acesso a todos os elementos, basta o **ponteiro para o primeiro elemento da lista. Este ponteiro nunca deve ser perdido** (caso aconteça perde-se acesso à lista).

Quando se começa a construir a lista, ela está vazia e, por isso, o ponteiro para o primeiro elemento é NULL.

Listas ligadas



Cada elemento desta lista é normalmente uma estrutura do tipo:

```
struct nome_do_tipo
{
    // Dados
    struct nome_do_tipo * seg; // ponteiro para o seguinte
};
```

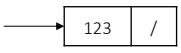

Listas ligadas

Inserir um elemento numa lista ligada:

O novo elemento deve ser criado antes de ser colocado na lista:

a. Alocar memória para o novo elemento (usando `malloc`)

b. Preencher o novo elemento:



1. colocar a NULL o ponteiro para o seguinte
2. atribuir valor aos restantes campos.

A inserção na lista depende da posição da lista onde se pretende colocar (início, fim, n-ésima posição, antes ou depois de determinado elemento,...).

Remover um elemento de uma listas ligadas

Se a lista está vazia, não há valor a remover.

Se o elemento a eliminar existe:

1. colocar um ponteiro a apontar para o elemento a remover
2. garantir que não é perdido o acesso à lista que está ligada ao elemento a remover através de ponteiros
3. libertar a memória ocupada pelo elemento a remover (usando `free`).

Listas ligadas

Usam-se quando não se sabe ou não se pretende definir o número de elementos de uma lista (não é possível usar tabelas)

cada vez que é inserido um novo elemento na lista é necessário reservar espaço na memória para ele

Funções da biblioteca `stdlib.h`

`void * malloc(size_t size)`

- pede ao sistema operativo um espaço de memória de tamanho `size`.
- recebe o número de bytes a alocar
- devolve um ponteiro para a zona de memória reservada ou NULL caso não seja possível alocar memória.

`void free(void * p)`

- liberta o espaço de memória (reservado pela malloc) apontado por p
- depois do espaço ser liberto deixa se ser possível aceder-lhe, é considerado pelo sistema operativo como memória disponível.

Listas ligadas

Considere as seguintes definições de tipos que permitem representar uma lista ligada linear simples de reais:


```
typedef struct lno * plista;
typedef struct lno
{
    float valor; // campo valor é um real
    plista prox; // campo prox é um ponteiro para um nó da lista
} listano;
```

ou

```
struct lno
{
    float valor;
    struct lno * prox;
};
typedef struct lno listano;
```

Listas ligadas

Exemplo

```c
typedef struct lno * plista;
typedef struct lno
{
     float valor; // campo valor é um real
     plista prox; // campo prox é um ponteiro para um nó da lista
} listano;

int main()
{
   // declara lista vazia
   plista plst = NULL;

   // coloca valor 12.3 no início da lista
   plista no = (plista)malloc(sizeof(listano));
   no->valor = 12.3;
   no->prox = plst;
   plst = no;

   // remove primeiro elemento da lista, considerando lista não vazia
   plista paux = plst;
   plst = plst->prox;
   free(paux);

    return 1;
}
```

Listas ligadas

Exemplo

(subprogramas)

Criar e devolver uma lista vazia

```c
plista crialista()
{
    plista p = NULL;
    return p;
}
```

Indicar se uma lista é vazia

```c
int listavazia(plista lst)
{
    if (lst == NULL)
        return 1;
    else
        return 0;
}
```

Listas ligadas

```
typedef struct lno * plista;
typedef struct lno
{
     float valor;
     plista prox;
} listano;
```

Exemplo (subprogramas)

Mostrar os elementos de uma lista

```
void escrevelista(plista lst)
{
    if(listavazia(lst))
        printf("a lista e vazia!!!\n");
    else
    {
        printf("[");
        do
        {
            printf(" %.1f,",lst->valor);
            lst = lst->prox;
        }while(lst != NULL);
        printf("\b ]\n");
    }
}
```

```
int main()
{
     plista pl = crialista();
     escrevelista(pl);

     return 1;
}
```


Listas ligadas

```
typedef struct lno * plista;
typedef struct lno
{
     float valor;
     plista prox;
} listano;
```

Exemplo (subprogramas)

Acrescentar um elemento x no início de uma lista

```
void junta_no_ini_lista(float x, plista * lst)
{
   plista no = (plista)malloc(sizeof(listano));

   if (no == NULL)
   {
        perror("ERRO!!! Nao ha memoria disponivel...");
        exit(-1);
   }
   else
   {
        no->valor = x;
        no->prox = * lst;
        * lst = no;
   }
}
```

```
int main()
{
     plista pl = crialista();
     escrevelista(pl);
     junta_no_ini_lista(1.2,&pl);
     escrevelista(pl);

     return 1;
}
```

Listas ligadas

```
typedef struct lno * plista;
typedef struct lno
{
    float valor;
    plista prox;
} listano;
```

Exemplo (subprogramas)

Acrescentar elemento no fim da lista

```
void junta_no_fim_lista(float x, plista * lst)
{
    plista pl = (plista)malloc(sizeof(listano));

    if (pl == NULL)
    {/* erro! */}
    else
    {
        pl->valor = x;
        pl->prox = NULL;
        if((* lst) == NULL)
            * lst = pl;
        else
        {
            plista paux = * lst;
            while (paux->prox != NULL)
                paux = paux->prox;
            paux->prox = pl;
        }
    }
}
```

Listas ligadas

```
typedef struct lno * plista;
typedef struct lno
{
    float valor;
    plista prox;
} listano;
```

Exemplo (subprogramas)

Remover 1º elemento da lista

```
void remove_prim(plista * lst)
{
    if (listavazia(*lst))
        printf("a lista já é vazia!!!\n");
    else
    {
        plista paux = * lst;
        *lst = (*lst)->prox;
        free(paux);
    }
}
```

estgoh
Politécnico de Coimbra

Listas ligadas

```
typedef struct lno * plista;
typedef struct lno
{
    float valor;
    plista prox;
} listano;
```

Exemplo

(subprogramas)

Remover último elemento da lista

```
void remove_ult(plista * lst)
{
    if (listavazia(*lst))
        printf("a lista já é vazia!!!\n");
    else
    {
        plista paux1, paux2;
        paux1 = * lst;
        paux2 = NULL;
        while(paux1->prox != NULL)
        {
            paux2 = paux1;
            paux1 = paux1->prox;
        }
        free(paux1);
        if(paux2 != NULL)
            paux2->prox = NULL;
        else * lst = NULL;
    }
}
```

estgoh
Politécnico de Coimbra

Listas ligadas

```
typedef struct lno * plista;
typedef struct lno
{
    float valor;
    plista prox;
} listano;
```

Exemplo

(subprogramas)

Remover n-esimo elemento da lista

```
void remove_nesimo (plista * lst, int n)
{
    if (n == 1)
        remove_prim(lst);
    else
    {
        plista paux1,paux2;
        int i = 1;
        paux1 = *lst;
        while(i < n-1 && paux1->prox != NULL)
        {
            paux1 = paux1->prox;
            i++;
        }
        if(paux1->prox == NULL)
            printf("lista nao tem elemento %i\n",n);
        else
        {
            paux2 = paux1->prox;
            paux1->prox = paux2->prox;
            free(paux2);
        }
    }
}
```