Alocação Dinâmica e Listas Encadeadas

Prof. Angelo Loula
UEFS

Alocação Dinâmica

- Muitas vezes usamos espaço desnecessário da memória declarando variáveis;
- Ou ainda ficamos limitados as variáveis declaradas.
- Alocação dinâmica (durante a execução do programa) de memória permite
  - Usar somente a memória necessária, diminuindo desperdício de memória;
  - Não impor limites quanto ao espaço na memória, a não ser físicos.

Alocação Dinâmica

- Funções da biblioteca `<stdlib.h>`
  - *malloc*
    - aloca espaço na memória e devolve um ponteiro (endereço) para ele
    - *malloc* cria somente um espaço e não inicializa

```
ponteiro = (tipo_de_dado *) malloc(tamanho_em_bytes);
```

  - *free*
    - libera espaço alocado na memória

```
free( ponteiro );
```

    - evita disperdício liberando o que não mais é usado

Alocação Dinâmica

```
#include <stdio.h>
#include <stdlib.h>
int main ( ){
 char *ptr ;
 ptr = (char *) malloc(sizeof(char));
 scanf("%c ", ptr) ;
 printf("%c \n", *ptr ) ;
 free(ptr) ;
}
```

Alocação Dinâmica

- *malloc*
  - **sizeof(***tipodedados***)**
    - indica quanto espaço um tipo de dados precisa
    - sizeof(char), sizeof(int), sizeof(float), sizeof(tipopessoa),…
  - conversão de tipo
    - malloc retorna um ponteiro sem tipo, 'genérico'
      - precisamos converter para o tipo necessário

```
ponteiro = (tipo_de_dado *) malloc(...
```


Alocação Dinâmica

```
#include <stdio.h>
#include <stdlib.h>
int main ( ){
 int *ptr ;
 int x ;
 ptr = (int *) malloc(sizeof(int)) ;
 scanf("%d %d" , ptr , &x ) ;
 *ptr = *ptr + x ;
 printf("%d\n", *ptr) ;
 free(ptr) ;
}
```


Alocação Dinâmica

```
#include <stdio.h>
#include <stdlib.h>
int main() {
 int *ptr;
 ptr = (int *) malloc (n * sizeof (int));
 if (ptr == NULL) {
    printf ("\nErro: nao foi possivel reservar
a memoria necessaria.\n");
 }
 else {
     /*ok - pode fazer o que há pra fazer */
 }
}
```


Alocação Dinâmica

```
#include <stdio.h>
#include <stdlib.h>
int main() {
 int *vetor, n, i;
 printf ("Quantos numeros deseja informar? ");
 scanf ("%d", &n);
 vetor = (int *) malloc (n * sizeof (int));
 for(i=0; i<n; i++){
   scanf("%d", &vetor[i]);
 }
 for(i=0; i<n; i++){
   printf(" %d", vetor[i]);
 }
}
```

Listas Encadeadas

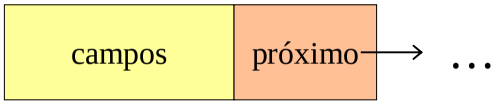
- Vetores nem sempre são ideais
  - Como variável seu tamanho é fixado antes do programa ser executado
  - Alocado dinamicamente, seu tamanho não pode aumentar mais!
- Listas Encadeadas
  - Criamos espaço individualemente para cada elemento que queremos guardar
  - Removemos espaço individualmente dos elementos que não queremos mais

Listas Encadeadas

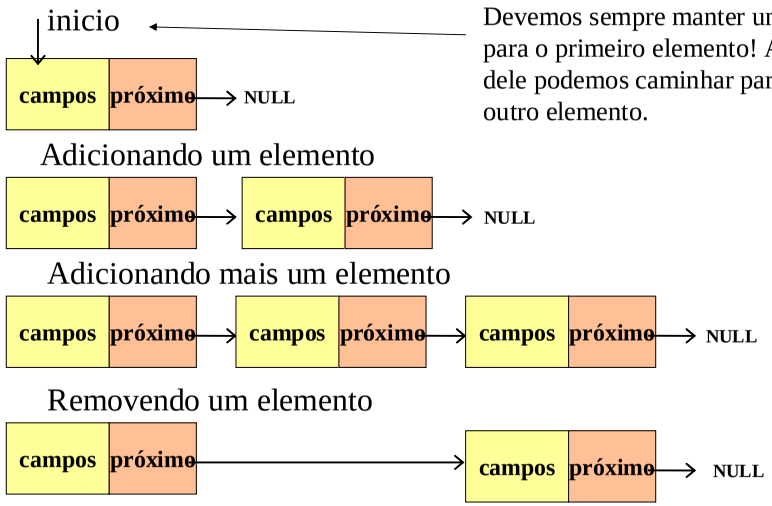
```
typedef struct elemento{
     int campo1;
     char campo2;
     struct elemento *proximo;
} elemento;
```

É preciso dar um nome antes para poder usá-lo antes que a definição acabe.

Endereço para o próximo elemento da lista



Listas Encadeadas



Devemos sempre manter um ponteiro para o primeiro elemento! Através dele podemos caminhar para qualquer outro elemento.

Listas Encadeadas

- Declarando uma lista encadeada

```
typedef struct no{
  int info;
  struct no *proximo;
} elemento;

int main(){
  elemento *iniciolista;
```

- Quando a lista está vazia, o primeiro elemento é NULL. Como toda lista começa vazia, inicializamos assim.

```
int main(){
  elemento *iniciolista;
  iniciolista=NULL;
```

Listas Encadeadas

- Caminhando na lista

```
p = iniciolista;
while (p!=NULL){
  printf(" %d", p->info); /* ou p=(*p).info*/
  p=p->proximo; /* ou p=(*p).proximo*/
}
```

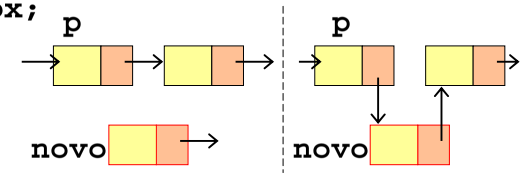
- Busca na lista

```
p = iniciolista ;
while ( (p!= NULL) && (p->info!= x)){
     p = p->proximo ;
}
if(p!=NULL){
     print("Encontrei!");
}
```


Listas Encadeadas

- Inserir na lista após um elemento p

```
novo = (elemento *)malloc(sizeof(elemento));
novo->info = y;
novo->prox = p->prox;
p->prox = novo;
```



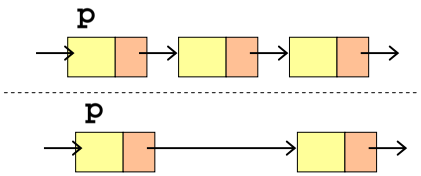
- Inserir no começo da lista

```
novo = (elemento *)malloc(sizeof(elemento));
novo->info = y;
novo->prox = iniciolista;
iniciolista = novo;
```


Listas Encadeadas

- Remover elemento após um elemento p

```
apaga = p->prox ;
p->prox = apaga->prox ;
free(apaga) ;
```



- Remover o primeiro da lista

```
apaga=iniciolista;
iniciolista= iniciolista->prox ;
free(apaga);
```


Listas Encadeadas

- Exercício: Função para
  - Dado um valor a ser inserido, inserí-lo ao final da lista
    - Caminhar e depois inserção
  - Dado um valor a ser inserido, inserí-lo antes do primeiro elemento que tenha valor maior que este. (Inserção ordenada)
    - Busca e depois inserção
  - Dado um valor a ser removido, remove-lo da lista se existir.
    - Busca e depois remoção