# Revisão P2 de Estruturas de Dados Dênio Duarte

Erickson G. Müller

13 de maio de 2024

## 1 Conteúdos

- 1. Alocação Dinâmica de Memória
- 2. Lista Simplesmente Encadeada
- 3. Lista Duplamente Encadeada
- 4. Filas e Pilhas

### 2 Alocação Dinâmica de Memória

A Alocação Dinâmica de Lista Encadeada na Memória segue os seguintes passos:

- 1. Alocação de memória
- 2. Colocar os valores no espaço alocado e NULLificar as variáveis ponteiras
- 3. Encadeamento

### 3 Estudo de Código

Desenvolver uma lista encadeada com alocação dinâmica é uma receita de bolo, para isso, iremos analisar cada etapa do código abaixo, que imprime uma sequência de pontos cartesianos.

```
#include <stdio.h>
#include <stdlib.h>
struct tpoint{
         struct tpoint *next;
typedef struct tpoint tpnt;
int main(){
         tpnt *p, *aux, *first = NULL;
         for (i=1; i \le 10; i++)
                  p = (tpnt*)malloc(sizeof(tpnt));
                  p\rightarrow x=i;
                  p->y=i+10;
                   if (first = NULL) {
                            first = p;
                            aux = p;
                   else{
                            aux -> next = p;
                            aux = p;
         for (aux=first; aux!=NULL; aux=aux->next){
                   printf("(\%d,\%d)\n", aux->x,aux->y);
         if (first!=\overline{NULL}){
                   aux=first;
                   while (aux->next!=NULL) {
                            aux = aux -> next;
                            free(p);
                   free (aux);
                   first=NULL;
```

#### 4 Definir a Estrutura

```
struct tpoint{
    int x,y;
    struct tpoint *next;
};
typedef struct tpoint tpnt;
```

- 1. Variáveis usadas nos itens dentro da lista
- Ponteiro \*next, que apontará para o próximo da lista
- 3. Atribuir um apelido para essa struct

### 5 free\_memory & $*ins\_end$

Antes de criarmos a função main, podemos inserir no código uma função para liberar o espaço de memória ou para fazer modificações na lista encadeada, como incluir novo item no meio da lista. Essas duas funções não estão presentes neste código e serão apresentadas ao final da explicação.

#### 6 Criar Variáveis Ponteiros

- 1. \*p aponta para o item
- $2.\ ^*{\rm aux}$  aponta para várias coisas ao mesmo tempo
- 3. \*first se for NULL (first = p)

### 7 Criar um Loop de Encadeamento

```
int i;
for(i=1;i<=10;i++){
    p = (tpnt *)malloc(sizeof(tpnt)):

    p->x=i;
    p->y=i+10;

    if(first == NULL){
        first = p;
        aux = p;
    }
    else{
        aux->next = p;
        aux = p;
    }
}
```

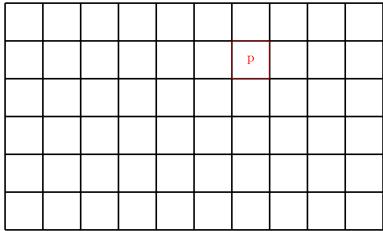
A primeira ação do loop é alocar espaço na memória através da linha:

Essa função vai designar um único espaço na memória para alterar suas variáveis conforme a iteração no loop. Assim, o programa fica mais leve na memória primária.

Lembrando as três etapas do encadeamento:

- 1. Alocação de memória
- 2. Colocar os valores no espaço alocado e NULLificar as variáveis ponteiras
- 3. Encadeamento





2 3

1 4