Revisão P2 de Estruturas de Dados
Dênio Duarte

Erickson G. Müller

May 13, 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.

```c
#include <stdio.h>
#include <stdlib.h>

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

int main(){
        tpnt *p, *aux, *first = NULL;
        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;
                }
        }

        for(aux=first;aux!=NULL;aux=aux->next){
                printf("(%d,%d)\n", aux->x,aux->y);
        }

        if(first!=NULL){
                aux=first;
                while(aux->next!=NULL){
                        p = aux;
                        aux = aux->next;
                        free(p);
                }
                free(aux);
                first=NULL;
        }
        return 0;
}
```

4 Definir a Estrutura

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

1. Variáveis usadas nos itens dentro da lista
2. 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

```
int(main){
        tpnt *p;
        tpnt *aux;
        tpnt *first = NULL;
```

1. *p - aponta para o item
2. *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:

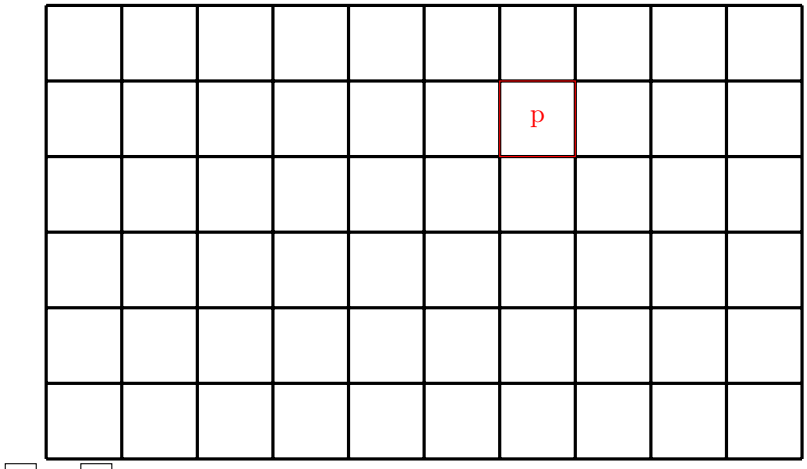
```
p = (tpnt *)malloc(sizeof(tpnt)
```

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

RAM



O código abaixo serve para atribuir valores às variáveis atuais. Pode ser de diversas formas diferentes.

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

Em seguida vamos encadear a lista:

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

O primeiro bloco é chamado quando a lista está vazia. Ele aponta o elemento p como first e o aux apontando para o próprio elemento. Inicialmente, todos apontam para a mesma região alocada.
Dentro do else, aux deve apontar sempre para a **região anterior** àquela a ser alocada, or isso que $aux->next = p$ antes de $aux = p$.

7.1 Como está a variável p em cada linha do loop

```
1 for(i=1;i<=10;i++){
2         p = (tpnt*)malloc(sizeof(tpnt));
3
4         p->x=i;
5         p->y=i+10;
6
7         if(first==NULL){
8                 first = p;
9                 aux = p;
10        }
11        else{
12                aux->next = p;
13                aux = p;
14        }
15 }
```

8 Função para Printar as Coordenadas

```
        for(aux=first; aux != NULL; aux = aux->nex){
                printf("(%d,%d)\n", aux->x,aux->y):
        }
```