Aula 07 - Estrutura de dados 1 – Uniube

Prof. Marcos Lopes

34 9 9878 0925

Lista encadeada (com a célula de cabeça)

Vamos revisar as operações até o momento numa lista com a célula da cabeça, analisando o comportamento do seguinte programa “lista_encadeada_com_cabeca.cpp”:

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

typedef struct celula celula;

struct celula {
    int conteudo;
    celula *prox;
};

void imprime(celula *le) {
   celula *p;
   for (p = le->prox; p != NULL; p = p->prox)
      printf ("%d\n", p->conteudo);
}

//continua
```

Lista encadeada (com a célula de cabeça)

```
celula* busca(int x, celula *le) {
    celula* p;
    p = le->prox;
    while(p != NULL && p->conteudo != x) {
        p = p->prox;
    }
    return p;
}

void insere(int x, celula *p){
   celula *nova;
   nova = (celula*) malloc (sizeof (celula));
   nova->conteudo = x;
   nova->prox = NULL;
   celula *current = p;
   while(current->prox != NULL) {
      current = current->prox;
   }
   current->prox = nova;
}

//continua
```

Lista encadeada (com a célula de cabeça)

```
void remove(int x, celula *p) {

    if(p->prox == NULL) { return; }

    struct celula* current = p->prox;
    struct celula* prev = p;

    while (current != NULL && current->conteudo != x) {
        prev = current;
        current = current->prox;
    }

    if (current == NULL) {
        return;
    }

    prev->prox = current->prox;
    free(current);
}

//continua
```

Lista encadeada (com a célula de cabeça)

```
int main(void) {

    celula *lista = NULL;
    lista = (celula*) malloc(sizeof(celula));   //head
    lista->conteudo = -999999;    lista->prox = NULL;

    insere(10, lista);
    insere(20, lista);
    insere(30, lista);
    insere(40, lista);
    insere(50, lista);

    remove(30, lista);
    insere(70, lista);
    imprime(lista);

    celula* achada = busca(40, lista);
    if(achada != NULL) {
        printf("Encontrou: %d\n\n", achada->conteudo);
    } else {
        printf("Não encontrou.");
    }

    return EXIT_SUCCESS;
}
```

Exercícios 1:

a) Testar o programa lista_encadeada_com_cabeca.cpp e verificar se todas as funções estão corretas

b) Implementar uma função que substitui um valor na lista, ou seja:

```
void substitui(int x, int novo, celula* le) {
	//implementação
}
```

Lista encadeada (sem a célula de cabeça)

Vamos revisar as operações até o momento numa lista sem a célula da cabeça, analisando o comportamento do seguinte programa “lista_encadeada_sem_cabeca.cpp”:

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

typedef struct celula celula;

struct celula {
    int conteudo;
    celula *prox;
};

void imprime(celula *le) {
   celula *p;
   for (p = le; p != NULL; p = p->prox)
      printf ("%d\n", p->conteudo);
}

//continua
```

Lista encadeada (sem a célula de cabeça)

```
celula* busca(int x, celula *le) {
    celula* p;
    p = le;
    while(p != NULL && p->conteudo != x) {
        p = p->prox;
    }
    return p;
}

void insere(int x, celula **p){
  celula *nova;
  nova = (celula*) malloc (sizeof (celula));
  nova->conteudo = x;
  nova->prox = NULL;
  if(*p == NULL) {
      *p = nova;
      return;
  }
  celula *current = *p;
  while(current->prox != NULL) {
      current = current->prox;
  }
  current->prox = nova;
}

//continua
```

Lista encadeada (sem a célula de cabeça)

```
void remove(int x, celula **p) {

    struct celula* current = *p;
    struct celula* prev = NULL;

    if (current != NULL && current->conteudo == x) {
        *p = current->prox;
        free(current);
        return;
    }

    while (current != NULL && current->conteudo != x) {
        prev = current;
        current = current->prox;
    }

    if (current == NULL) {
        return;
    }

    prev->prox = current->prox;
    free(current);
}

//continua
```

Lista encadeada (sem a célula de cabeça)

```
int main(void) {

    celula *lista = NULL;

    insere(10, &lista);
    insere(20, &lista);
    insere(30, &lista);
    insere(40, &lista);
    insere(50, &lista);

    remove(30, &lista);
    insere(70, &lista);

    imprime(lista);

    celula* achada = busca(40, lista);
    if(achada != NULL) {
        printf("Encontrou: %d\n\n", achada->conteudo);
    } else {
        printf("Não encontrou.\n\n");
    }

    return EXIT_SUCCESS;
}
```

Exercícios 2:

a) Testar o programa lista_encadeada_sem_cabeca.cpp e verificar se todas as funções estão corretas

b) Implementar uma função que substitui um valor na lista, ou seja:

```
void substitui(int x, int novo, celula* le) {
	//implementação
}
```

Questão1: é necessário o uso de ponteiro de ponteiro?

Questão2: caso não seja necessário o uso de ponteiro de ponteiro, ainda assim posso implementar usando o recurso de ponteiro de ponteiro?