Manipulações em Listas Encadeadas

Lucas Vinícius de Lima Assis
(62) 9 9973-7345

17 de dezembro de 2023

Considerações iniciais

"Este é um pequeno guia para treinar a leitura de código e revisar algumas manipulações com listas encadeadas. Eu não sei as questões da prova, mas tenho certeza de que estar familiarizado com essas operações elementares ao ponto de ler um código e entender o que ele faz com a lista, faz parte de um bom preparo. Enfase para o faz parte, porque ainda existe muito mais a ser explorado que eu não consigo trazer aqui neste pequeno documento. Sem mais delongas, bons estudos, e qualquer dúvida me chamem no zap, eu tenho o gabarito."

-O Monitor

Identificar operações

Eu andei programando um pouco esses dias e implementei uma lista simplesmente encadeada e algumas operações para mexer com ela. Vou colocar alguns dos códigos das funções para vocês identificarem o que cada função faz. Já vou dizendo que há algumas que não funcionam para todos os casos, fiquem atentos a isso também e identifiquem as que não funcionam para qualquer caso. E o principal: **todas possuem um significado, não são operações aleatórias**.

Para todas as funções, eu utilizei essa declaração de Lista/Nó. A lista no caso é um ponteiro para o primeiro Nó.

```
// Declaração da Lista
typedef struct No {
    int info;
    struct No *prox;
} No;
```

```
// Função A
No* funcaoA() {
    return NULL;
}

// Exemplo de uso
int main() {
    No* lista = funcaoA()
    ...
}
```

Sobre a próxima função, por que eu usei `No** plista` e não `No* lista`? Eu poderia fazer do segundo jeito?

```
// Função B
void funcaoB(No** plista, int v) {
    No* n = (No*) malloc(sizeof(No));
    n->info = v;
    n->prox = (*plista);
    (*plista) = n;
}

// Exemplo de uso
int main () {
    No *lista = funcaoA();
    funcaoB(&lista, 4);
    funcaoB(&lista, 12);
    ...
}
```

As funções C_a e C_b pretendem fazer a mesma coisa, mas de maneiras diferentes. Ambas funcionam? Qual a diferença entre elas?

```
// Função Ca
void funcaoCa(No* lista) {
    No* aux = lista;
    while (aux != NULL) {
        printf("(%d)->", aux->info);
        aux = aux->prox;
    }
    printf("NULL\n");
}
```

```
// Função Cb
void funcaoCb(No* lista) {
    while (lista != NULL) {
        printf("(%d)->", lista->info);
        lista = lista->prox;
    }
    printf("NULL\n");
}
```

```
// Função D
void funcaoD(No **plista) {
    No *aux = (*plista);
    No *l = (*plista);

    while (l != NULL) {
        l = l->prox;
        free(aux);
        aux = l;
    }
    (*plista) = NULL;
}
```

```
// Função E
void funcaoE(No **plista, int valor) {
    No *n = (No*) malloc(sizeof(No));
    n->info = valor;

    No *l = (*plista);
    No *p = l->prox;

    while (p != NULL && p->info < valor) {
        l = l->prox;
        p = p->prox;
    }

    n->prox = p;
    l->prox = n;
}
```

```
// Função F
void funcaoF(No* lista) {
    No *a, *b;

    int t = 1;
    while (t) {
        a = lista;
        b = a->prox;
        t = 0;
        while (b != NULL) {
            if (a->info > b->info) {
                int aux = a->info;
                a->info = b->info;
                b->info = aux;
                t = 1;
            }
            a = b;
            b = b->prox;
        }
    }
}
```

```
// Função G
void funcaoG(No **plista, int n, int l) {
    for (int i=0; i<n; i++) {
        funcaoB(plista, rand()%l);
    }
}
```

```
// Função H
No* funcaoH(No *lista, int n) {
    while (lista != NULL && lista->info != n) {
        lista = lista->prox;
    }
    return lista;
}
```

```
// Função Ia
void funcaoIa(No** plista) {
    No *a, *b, *c;
    a = (*plista);
    b = a->prox;
    c = NULL;
    while (a != NULL) {
        a->prox = c;
        c = a;
        a = b;
        if (b != NULL)
            b = b->prox;
    }
    (*plista) = c;
}
```

```
// Função Ib
void funcaoIb(No** plista, No* a, No* b, No* c) {
    a->prox = c;
    c = a;
    a = b;
    if (b != NULL)
        b = b->prox;

    if (a != NULL) {
        funcaoIb(plista, a, b, c);
    } else {
        (*plista) = c;
    }
}

// Exemplo de uso
int main() {
    ...
    No* lista = funcaoA();
    funcaoG(&lista, 20, 10);
    funcaoIb(&lista, lista, lista->prox, NULL);
}
```