



**INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA MINAS
GERAIS**

BACHARELADO EM SISTEMAS DE INFORMAÇÃO

BIANCA DE CASTRO AGUIAR FONTES

ATIVIDADE 8

Lista dinâmica

SÃO JOÃO EVANGELISTA

2023

SUMÁRIO

1. FUNCIONALIDADE DE UMA LISTA COM ALOCAÇÃO DINÂMICA EM C++	3
2. CÓDIGO COMENTADO	4
a. Bibliotecas e Namespace	4
b. Estrutura de dados "Item"	4
c. Estrutura da lista	5
d. Ponteiro do tipo lista como nulo e função para criar lista vazia	5
e. Função que verifica se a lista está vazia	5
f. Função insere item na última posição	6
g. Função que mostra os itens da lista	6
h. Função insere item na primeira posição da lista	7
i. Função que insere item em uma posição específica	8
j. Função que remove o primeiro item da lista	9
k. Função que remove o último item da lista	10
l. Função que remove o item de uma posição específica	11
m. Função menu	12
n. Função main	13

1. FUNCIONALIDADE DE UMA LISTA COM ALOCAÇÃO DINÂMICA EM C++

A lista é uma estrutura de dados onde seus itens são organizados de forma automática durante a execução. Podemos inserir, remover e editar itens mantendo a lista organizada utilizando ponteiros que indicam o próximo item.


Para que uma lista com alocação dinâmica funcione em um programa desenvolvido em c++, precisamos de quatro ponteiros:

- próximo – aponta sempre o próximo item da lista. (tipo de estrutura item)
- primeiro – item cabeça, responsável por apontar sempre o primeiro item da lista. (tipo de estrutura item)
- Último – responsável por apontar último item da lista (tipo de estrutura item)
- L – que é de fato a lista dinâmica (tipo de estrutura lista)

Ao inserir um elemento na lista dinâmica, o valor do ponteiro próximo do último item é atualizado para apontar para o novo elemento, expandindo assim a lista. Um processo semelhante ocorre ao remover um elemento, onde o ponteiro próximo do penúltimo item passa a apontar para o item nulo, liberando assim o espaço de memória do elemento removido com o delete na variável auxiliar. Esses processos são realizados de maneira coordenada pelos ponteiros mencionados para inserção e remoção em qualquer posição da lista, possibilitando a manipulação dinâmica.


2. CÓDIGO COMENTADO

a. Bibliotecas e Namespace



```
1  #include <iostream>
2  #include <string>
3
4  using namespace std;
```

b. Estrutura de dados “Item”



```
1  // estrutura do item
2  struct Item
3  {
4      int idade;
5      string nome;
6      Item *proximo; // variável que aponta
                       para o próximo item
7  };
```

c. Estrutura da lista

```

1 // estrutura da lista encadeada
2 struct Lista
3 {
4     Item *primeiro = NULL; // criando ponteiro como nulo
    (primeiro item)
5     Item *ultimo = NULL; // criando ponteiro como nulo
    (último item)
6 };

```

d. Ponteiro do tipo lista como nulo e função para criar lista vazia

```

1 // cria um ponteiro L do tipo lista, como nulo
2 Lista *L = NULL;
3
4 // cria uma lista vazia utilizando ponteiro
5 void criarLista()
6 {
7     L->primeiro = new Item; // cria um item como primeiro
8     L->ultimo = L->primeiro; // o ponteiro último recebe o
    ponteiro primeiro
9     L->primeiro->proximo = NULL; // cria o próximo item co
    mo nulo
10 }

```

e. Função que verifica se a lista está vazia

```

1 // retorna 0 ou 1, sendo 0 para primeiro diferente do
    último e 1 para primeiro igual ao último
2 int vazia()
3 {
4     return (L->primeiro == L->ultimo);
5 }

```

f. Função insere item na última posição

```

1 // insere um item na lista na última posição
2 void inserirUltima()
3 {
4     Item *x = new Item; // cria um item x
5
6     cout << "Digite um nome: ";
7     cin >> x->nome; // recebe o nome
8     cout << "Digite a idade: ";
9     cin >> x->idade; // recebe a idade
10
11     L->ultimo->proximo = x; // aponta para o último
    item e faz com que o próximo receba x
12     L->ultimo = x; // x passa a ser o último
13     L->ultimo->proximo = NULL; // o próximo item é
    definido como nulo
14 }

```

g. Função que mostra os itens da lista

```

1 // mostra os itens da lista
2 void mostrar()
3 {
4     Item *aux; // cria um item auxiliar
5     aux = L->primeiro->proximo; // atribui o próximo
    para a variável auxiliar
6     while (aux != NULL) // enquanto o próximo não for
    nulo, exibe os valores
7     {
8         cout << "Nome: " << aux->nome << " "
9         << "Idade: " << aux->idade << endl;
10        aux = aux->proximo; // aux recebe o próximo i
    tem
11    }
12 }

```

h. Função insere item na primeira posição da lista

```
1 // insere um item na primeira posição
2 void inserirPrimeira()
3 {
4     Item *x = new Item; // cria um item x
5
6     cout << "Digite um nome: ";
7     cin >> x->nome; // recebe o nome
8     cout << "Digite a idade: ";
9     cin >> x->idade; // recebe a idade
10
11     if (!vazia()) // se a lista não estiver vazia
12     {
13         x->proximo = L->primeiro->proximo; // o próximo
14         // do item x, recebe o primeiro item da lista
15         L->primeiro->proximo = x; // o primeiro item da
16         // lista recebe o item x
17     }
18     else // se a lista estiver vazia
19     {
20         L->ultimo->proximo = x; // o próximo item depois
21         // do último recebe x
22         L->ultimo = x; // o último item recebe x
23         L->ultimo->proximo = NULL; // o próximo item de
24         // pois do último recebe nulo
25     }
26 }
```

i. Função que insere item em uma posição específica

```
1 // insere um item em uma determinada posição informada p  
  ela usuário  
2 void inserirPosicao(int n)  
3 {  
4     int i = 0;  
5     Item *aux = L->primeiro; // variável auxiliar que re  
      ceber o primeiro item da lista  
6     while (i < (n - 1) && aux != NULL) // enquanto i for  
      menor que o a posição inserida - 1 e aux não for nulo  
7     {  
8         i++; // i acrescenta de um em um  
9         aux = aux->proximo; // aux recebe o próximo item  
10    }  
11  
12    Item *x = new Item; // cria um item x para receber o  
      novo item  
13    cout << "Digite um nome: ";  
14    cin >> x->nome; // recebe o nome  
15    cout << "Digite a idade: ";  
16    cin >> x->idade; // recebe a idade  
17  
18    x->proximo = aux->proximo; // o próximo item de x re  
      cebe o próximo item da lista  
19    aux->proximo = x; // o próximo item da lista recebe  
      o novo item  
20 }
```


j. Função que remove o primeiro item da lista

```
1 // remove o primeiro item da lista
2 void removerPrimeira()
3 {
4     if (!vazia()) // se a lista não estiver vazia
5     {
6         Item *aux = L->primeiro->proximo; // cria uma va
riável atribuindo o primeiro item da lista
7         cout << endl
8             << "+++ REMOVENDO +++" << endl;
9         cout << aux->nome << " " << aux->idade << endl
10            << endl; // mostra os valores do item a ser
removido
11         L->primeiro->proximo = aux->proximo; // atribui
o item depois de aux ao primeiro item da lista
12         if (aux == L->ultimo) // se aux for o último ite
m da lista
13         {
14             L->ultimo = L->primeiro; // último recebe o
item cabeça
15         }
16         delete aux; // deleta o o valor de aux
17     }
18     else // se não
19     {
20         cout << endl
21             << "Vazia" << endl; // imprime vazia
22     }
23 }
```

k. Função que remove o último item da lista

```
1 // remove o último item da lista
2 void removerUltima()
3 {
4     if (!vazia()) // se a lista não estiver vazia
5     {
6         Item *aux = L->primeiro; // cria uma variável qu
e recebe o item cabeça
7         while (aux->proximo != L->ultimo) // enquanto o
próximo item não for o último
8         {
9             aux = aux->proximo; // aux recebe o próximo
item
10        }
11        cout << endl
12             << "+++ REMOVENDO +++" << endl;
13        cout << aux->proximo->nome << " " << aux->proxim
o->idade << endl
14             << endl; // imprime o item a ser removido
15        aux->proximo = NULL; // o próximo item recebe nu
lo
16        delete L->ultimo; // deleta o último item da lis
ta
17        L->ultimo = aux; // o último recebe aux
18    }
19    else // se não
20    {
21        cout << endl
22             << "Vazia" << endl; // imprime vazia
23    }
24 }
```

I. Função que remove o item de uma posição específica

```

1 // remove o item de uma posição determinada pelo usuário
2 void removerPosicao(int posicao)
3 {
4     if (!vazia()) // se a lista não estiver vazia
5     {
6         int i = 1;
7         Item *aux1 = L->primeiro, *aux2 = L->primeiro->proximo; // duas
// variáveis, uma aponta recebe o item cabeça e a outro o primeiro item da
// lista
8         while (i < posicao && aux2 != NULL) // enquanto i for menor que
a posição inserida e o primeiro item da lista não for nulo
9         {
10            i++; // i acrescenta de um em um
11            aux1 = aux2; // o item cabeça recebe o primeiro item da list
a não for nulo
12            aux2 = aux2->proximo; // o primeiro item da lista recebe o p
róximo
13        }
14        if (aux2 == NULL) // se o primeiro item da lista for nulo
15        {
16            cout << "Nada para remover" << endl;
17        }
18        else // se não
19        {
20            cout << endl
21                << "+++ REMOVENDO +++" << endl;
22            cout << aux2->nome << " " << aux2->idade << endl
23                << endl; // exibe o item a ser removido
24            if (aux2 == L->ultimo) // se o primeiro item da lista for o
último
25            {
26                L->ultimo = aux1; // último recebe o item cabeça
27            }
28            aux1->proximo = aux2->proximo; // o item cabeça recebe o pró
ximo item
29            delete aux2; // deleta os valores de aux2
30        }
31    }
32 }

```

m. Função menu

```
1 // exibe o menu e retorna a opção inserida
2 int menu()
3 {
4     int opcao;
5     cout << "++++ Opcoes ++++" << endl;
6     cout << "1. Inserir no inicio" << endl;
7     cout << "2. Inserir no final" << endl;
8     cout << "3. Inserir em uma posicao" << endl;
9     cout << "4. Mostrar" << endl;
10    cout << "5. Remover primeira" << endl;
11    cout << "6. Remover ultima" << endl;
12    cout << "7. Remover de uma posicao" << endl;
13
14    cout << "0. Sair" << endl;
15    cout << "Digite: ";
16    cin >> opcao;
17    return opcao;
18 }
```

n. Função main

```
1 // executa o programa com base nas funções criadas e o opção informada
  pelo usuário
2 int main()
3 {
4     int opcao, p; // cria variáveis úteis para executar o menu e recebe
  r posições da lista, informa pelo usuário
5     L = new Lista; // cria uma lista
6
7     criarLista();
8     do
9     {
10         opcao = menu();
11         switch (opcao)
12         {
13             case 1:
14                 inserirPrimeira();
15                 break;
16             case 2:
17                 inserirUltima();
18                 break;
19             case 3:
20                 cout << "Digite a posicao: ";
21                 cin >> p;
22                 inserirPosicao(p);
23                 break;
24             case 4:
25                 mostrar();
26                 break;
27             case 5:
28                 removerPrimeira();
29                 break;
30             case 6:
31                 removerUltima();
32                 break;
33             case 7:
34                 cout << "Digite a posicao: ";
35                 cin >> p;
36                 removerPosicao(p);
37                 break;
38             case 0:
39                 cout << "Saindo ..." << endl;
40                 break;
41             default:
42                 cout << "Selecione uma opção válida!" << endl;
43                 break;
44         }
45     } while (opcao != 0);
46     return 0;
47 }
```