



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

BACHARELADO EM SISTEMAS DE INFORMAÇÃO

BIANCA DE CASTRO AGUIAR FONTES

ATIVIDADE 7

Pilha dinâmica

SÃO JOÃO EVANGELISTA

2023

SUMÁRIO

1. FUNCIONALIDADE DE UMA PILHA COM ALOCAÇÃO DINÂMICA EM C++	3
2. CÓDIGO COMENTADO	4
a. Bibliotecas e Namespace	4
b. Estrutura de dados “Item”	4
c. Ponteiro “topo”	5
d. Cabeçalhos das funções	5
e. Função “main”	6
f. Função “menu”	7
g. Função “empilhar”	7
h. Função “verificarSeTemAlgumaCoisa”	8
i. Função “desempilhar”	8
j. Função “mostrar”	9

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

A pilha é uma estrutura de dados que segue uma regra onde o último elemento a ser inserido é o primeiro a ser removido, ou seja, os elementos são inseridos ou removidos do topo.

A alocação dinâmica consiste em alocar memória durante a execução do programa, ao contrário de declará-la no código. Se torna bem usual em exemplos onde o tamanho da estrutura de dados não é definida ou varia durante a execução do programa.


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

- topo – aponta sempre o último elemento inserido na pilha.
- aux – aponta sempre para o elemento anterior da fila, possibilitando assim uma ligação entre os elementos da pilha.

Ao inserir um elemento na pilha, o valor do ponteiro “topo” aumenta e aloca automaticamente a memória para o novo elemento, um processo parecido ocorre ao remover um elemento, já que o ponteiro “topo” diminui o valor e libera o espaço de memória que estava alocado para o elemento removido. Esses processos são realizados utilizando os ponteiros citados, onde se relacionam, recebem e passam valores entre si, possibilitando a criação da pilha.


2. CÓDIGO COMENTADO

a. Bibliotecas e Namespace



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

b. Estrutura de dados “Item”



```
1  // define a estrutura item, que contém um valor de idade,
   // um nome e um ponteiro para o próximo elemento da pilha
2  struct Item
3  {
4      int idade;
5      string nome;
6      Item *proximo;
7  };
```

c. Ponteiro “topo”

```
1 // topo é um ponteiro que rastreia o último item da  
  pilha, iniciado como NULL para representar "pilha va  
  zia"  
2 Item *topo = NULL;
```

d. Cabeçalhos das funções

```
1 // funções declaradas antes da main para que possam  
  ser utilizadas no código principal com mais organiza  
  ção  
2 void empilhar();  
3 void desempilhar();  
4 bool verificarSeTemAlgumaCoisa();  
5 void mostrar();  
6 int menu();
```

e. Função “main”

```
1 // a função main é o ponto de partida do programa com
  um loop Do While, onde é exibido um menu de opções pa
  ra o usuário escolher e executar o que deseja. O loop
  se repete até que o usuário escolha sair do programa
  com a opção (0)
2 int main()
3 {
4     int opcao;
5     do
6     {
7         opcao = menu();
8         switch (opcao)
9         {
10            case 1:
11                empilhar();
12                break;
13            case 2:
14                desempilhar();
15                break;
16            case 3:
17                mostrar();
18                break;
19            case 0:
20                cout << "Saindo ... " << endl;
21                break;
22            default:
23                cout << "Selecione uma opção válida!" <<
24                endl;
25                break;
26            }
27        } while (opcao != 0);
28        return 0;
29    }
```

f. Função “menu”

```
1 // exibe uma série de opções ao usuário e retorna a opção es
  colhida
2 int menu()
3 {
4     int opcao;
5     cout << "+++ Opções +++ " << endl;
6     cout << "1. Inserir" << endl;
7     cout << "2. Remover" << endl;
8     cout << "3. Mostrar" << endl;
9     cout << "0. Sair" << endl;
10    cout << "Digite: ";
11    cin >> opcao;
12    return opcao;
13 }
```

g. Função “empilhar”

```
1 // solicita um nome e uma idade para um novo elemento, coloc
  a esse elemento no topo da pilha e atualiza o "topo"
2 void empilhar()
3 {
4
5     Item *temp = new Item; // cria um ponteiro para um novo
  objeto na memória heap
6     cout << "Nome: ";
7     cin >> temp->nome; // o valor é inserido no campo nome d
  o objeto temp
8     cout << "Idade: ";
9     cin >> temp->idade; // o valor é inserido no campo idade
  do objeto temp
10    temp->proximo = topo; // atualiza o campo próximo do tem
  p, aponta o elemento que era o topo anteriormente criando um
  a ligação entre o novo elemento e o anterior
11    topo = temp; // após atualizar o campo próximo, atualiza
  o ponteiro "topo" para apontar o novo elemento, tornando ele
  o topo da pilha
12    temp = NULL; // define como NULL para não vazar memória
  já que o temp já foi alocado dinamicamente.
13 }
```

h. Função “verificarSeTemAlgumaCoisa”

```
1 // com base no ponteiro "topo" verifica se há algum elemento
  na pilha, retornando um valor booliano
2 bool verificarSeTemAlgumaCoisa()
3 {
4     if (topo != NULL)
5     {
6         return true;
7     }
8     return false;
9 }
```

i. Função “desempilhar”

```
1 // remove o último elemento da pilha após verificar se existem eleme
  ntos
2 void desempilhar()
3 {
4     if (verificarSeTemAlgumaCoisa())
5     {
6         Item *temp = topo; // o ponteiro temp recebe o topo
7         topo = topo->proximo; // o elemento anterior é atribuído ao
  topo, sobrescrevendo
8         cout << "Removido: ";
9         cout << temp->nome << " " << temp->idade << endl; // imprime
  o elemento que foi removido
10        delete temp; // delete o temp, liberando espaço na memória
11    }
12    else
13    {
14        cout << "Nada!" << endl;
15    }
16 }
```


j. Função “mostrar”

```
1 // exibe o último elemento da pilha enquanto o ponteiro temp for
  diferente de NULL
2 void mostrar()
3 {
4     Item *temp = topo;
5     while (temp != NULL)
6     {
7         cout << temp->nome << " " << temp->idade << endl;
8         temp = temp->proximo;
9     }
10 }
```