9
LIFO (Pilha)
com Arranjos

Estruturas Homogêneas

Permite o armazenamento de apenas um datatype por vez.

Exemplo: int x[10]; string y[5]; double z[2];

Estes vetores somente permitem o armazenamento dos datatypes que foram criados.

Estruturas Heterogêneas

Permite o armazenamento de vários datatypes, exemplo:

Struct x { int x[10]; string y[15]; } tab;

Um struct armazena vários datatypes em seus atributos.

Arranjos - Estruturas Limitadas

Os arranjos são estruturas que permite o armazenamento limitado de elementos, por isso possuem tamanho pre-definido, são estruturas que utilizam structs e vetores com tamanho definido.

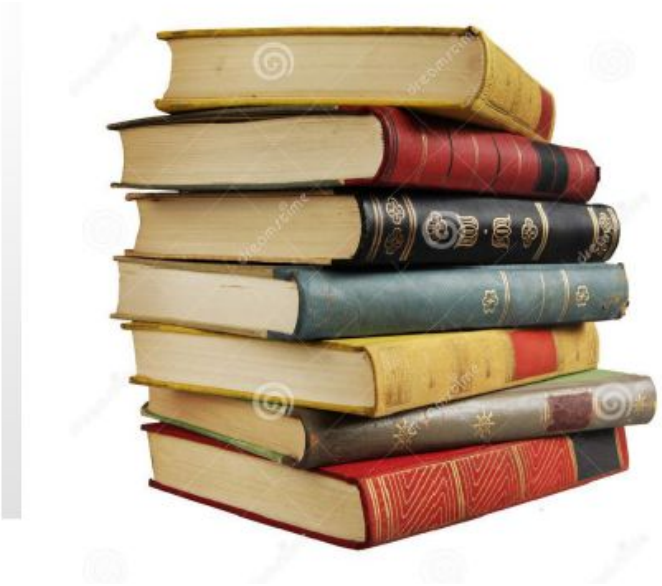
```
#define max 3
using namespace std;
struct mystr { int idade [max];
               string nome[max];  } reg;
```

Estruturas Dinâmicas

Estruturas dinâmicas não possuem limite de armazenamento, ou seja, o armazenamento dependerá da quantidade de memória disponível, neste caso, usa-se classes (POO) ou alocação dinâmica de memória envolvendo ponteiros.

LIFO - Last in First out

LIFO é uma estrutura do tipo pilha onde o último elemento que entra será o primeiro elemento a ser removido da pilha, como em uma pilha de livros ou pratos.



Atributos comuns da Pilha

TOPO: Esta variável irá sempre guardar a posição do último elemento a ser inserido na LIFO.

TAMANHO: Esta constante irá armazenar a quantidade máxima de elementos suportada pela LIFO do tipo arranjo simples.

DADO: Seria o elemento, número, texto ou registro a ser empilhado.

Sub-rotinas, funções da LIFO

PUSH(): Nome da função que empilha elementos

POP(): Nome da função que desempilha elementos

EXIBIR(): Nome da função que exibe a pilha na tela

VAZIA(): Função que checa se a LIFO está sem elementos

CHEIA():Função que checa se a LIFO está cheia e não cabe mais elementos.

Programa 1: LIFO com arranjo Simples - Parte 1

```
#include "iostream"
#define max 3
using namespace std;

typedef struct lifo pilha;

struct lifo
{ int topo;
 int dado[max]; };

int lerValor()
{ int valor;
 cout << "\nDigite o valor a ser empilhado:";
 cin >> valor;  return valor; }

bool pilhaCheia(pilha p)
{ if (p.topo == max - 1)
 return true;
 return false; }

bool pilhaVazia(pilha p)
{ if (p.topo==-1)
 return true;     return false; }
```

Programa 1 : LIFO com arranjo Simples - Parte 2

```
pilha push (int valor, pilha p) {
 if ( pilhaCheia ( p ) == true) {
     cout << "\nPilha Cheia\n";
     system("pause");
     return p; }

 p.topo ++;
 p.dado[p.topo] = valor;
 cout << "\nO valor empilhado foi" << valor << endl;
 system("pause");
 return p; }

pilha pop (pilha p) {
if (pilhaVazia(p) == true)
{ cout << "\nA pilha já está vazia!";
 system("pause"); return p; }

cout << "\nO valor desempilhado será:" << p.dado[p.topo] << endl;
system("pause");
p.dado[p.topo] = '\0';
p.topo --;   return p; }
```

Programa 1 : LIFO com arranjo Simples - Parte 3

```
void mostrarPilha (pilha p) {

if (pilhaVazia(p) == true)
{
   cout << "\nA pilha está vazia!";
   system("pause"); return; }

  for (int i=p.topo; i>=0; i--)
     cout << p.dado[i] << endl; system("pause");
}

int tela() {

int tecla;

system("cls");
cout << "\nMenu\n1 Push\n2 Pop\n3 Mostrar
pilha\n4 Sair\nItem:";
cin >> tecla;
return tecla;  }
```

Programa 1 : LIFO com arranjo Simples - Parte Final

```
void controlarPilha (pilha p){
int tecla, valor;

pilha p1;
p1.topo = -1;

do { tecla = tela();
   switch(tecla)    {
     case 1: valor = lerValor();
             p1 = push(valor, p1);
             break;

     case 2: p1 = pop(p1); break;

     case 3: mostrarPilha ( p1 );    }

} while (tecla != 4);
cout << "\nPrograma Finalizado...!\n"; }

int main()  {
  pilha p1;
 controlarPilha ( p1 );
 return 0; }
```

TAREFA / AVALIAÇÃO CONTINUADA

Faça um programa estruturado para controlar uma pilha com arranjo simples contendo os atributos: IDADE, NOME e SALÁRIO. O código da LIFO deverá conter as seguintes funções obrigatórias:

A. push() para empilhar o registro
B. pop() para remover registros
C. int mostrar() para mostrar os registros da LIFO
D. bool cheia() verificar se a LIFO está cheia
E. bool vazia() verificar se a LIFO está vazia
F. int tela () exibe a tela e armazena o valor da opção de menu selecionada
G. void Controle() controla o menu de controle da LIFO