Ponteiro para Ponteiro(Ponteiro Duplo)

Int x = 10

Int \*p = &x;

Int \*\*pp = &p

pp guarda o endereço de p, ou seja, guarda o valor de p

\*pp → acessa p.

\*\*pp → acessa o conteúdo de p, ou seja, o valor de x.

|  |  |  |
| --- | --- | --- |
| **Pilha (Stack)** – LIFO: Last In, First Out – Conceito Inicial**Operação** | **O que faz** |  |
| push | Insere elemento no topo da pilha |  |
| pop | Remove e retorna o valor do topo |  |
| peek ou top | Retorna o valor do topo **sem remover** |  |

**Implementando Exemplo 1:**

#include <stdio.h>

#include <stdlib.h>

// Definição da estrutura do nó da pilha

struct No {

int valor;

struct No \*prox;

};

// Função push: insere um novo elemento no topo da pilha

void push(struct No \*\*topo, int valor) {

struct No \*novo = (struct No \*)malloc(sizeof(struct No)); // Aloca novo nó

novo->valor = valor; // Atribui o valor

novo->prox = \*topo; // Faz o novo nó apontar para o antigo topo

\*topo = novo; // Atualiza o topo da pilha

}

// Função pop: remove o topo da pilha e retorna seu valor

int pop(struct No \*\*topo) {

if (\*topo == NULL) {

printf("Pilha vazia!\n");

return -1; // Valor que indica erro

}

struct No \*temp = \*topo; // Armazena o nó atual

int valor = temp->valor; // Pega o valor antes de remover

\*topo = temp->prox; // Atualiza o topo para o próximo nó

free(temp); // Libera a memória do nó antigo

return valor; // Retorna o valor removido

}

// Função peek: retorna o valor do topo da pilha sem remover

int peek(struct No \*topo) {

if (topo == NULL) {

printf("Pilha vazia!\n");

return -1; // Valor sinalizador

}

return topo->valor; // Retorna o valor do topo

}

// Função para imprimir a pilha (visualização)

void imprimirPilha(struct No \*topo) {

printf("Topo da pilha\n");

while (topo != NULL) {

printf(" %d\n", topo->valor);

topo = topo->prox;

}

printf("NULL\n");

}

int main() {

struct No \*pilha = NULL; // Pilha vazia

// Inserções

push(&pilha, 10);

push(&pilha, 20);

push(&pilha, 30);

imprimirPilha(pilha); // Mostra a pilha

// Consulta o topo sem remover

printf("Valor no topo (peek): %d\n", peek(pilha));

// Remove o topo

int valorRemovido = pop(&pilha);

printf("Valor removido (pop): %d\n", valorRemovido);

// Consulta novamente após remoção

printf("Novo topo (peek): %d\n", peek(pilha));

imprimirPilha(pilha); // Mostra a pilha final

return 0;

}

**Exemplo 2:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

// Estrutura para representar uma página

struct Pagina {

char url[100];

struct Pagina \*prox;

};

// Função para acessar uma nova página (push)

void acessarPagina(struct Pagina \*\*topo, const char \*endereco) {

struct Pagina \*nova = (struct Pagina \*)malloc(sizeof(struct Pagina));

strcpy(nova->url, endereco); // Copia o endereço da nova página

nova->prox = \*topo; // Aponta para a página anterior

\*topo = nova; // Atualiza o topo da pilha

}

// Função para voltar para a página anterior (pop)

void voltarPagina(struct Pagina \*\*topo) {

if (\*topo == NULL) {

printf("Nenhuma página para voltar.\n");

return;

}

struct Pagina \*temp = \*topo;

printf("Voltando da página: %s\n", temp->url);

\*topo = temp->prox; // Atualiza o topo

free(temp); // Libera o nó removido

}

// Função para mostrar a página atual

void mostrarPaginaAtual(struct Pagina \*topo) {

if (topo == NULL) {

printf("Nenhuma página aberta.\n");

} else {

printf("Página atual: %s\n", topo->url);

}

}

// Função para mostrar o histórico (todos os nós da pilha)

void mostrarHistorico(struct Pagina \*topo) {

if (topo == NULL) {

printf("Histórico vazio.\n");

return;

}

printf("Histórico de páginas visitadas:\n");

while (topo != NULL) {

printf(" - %s\n", topo->url);

topo = topo->prox;

}

}

// Função principal com menu interativo

int main() {

struct Pagina \*pilha = NULL; // Pilha de páginas

int opcao;

char url[100];

do {

printf("\n==== MENU DO NAVEGADOR ====\n");

printf("1. Acessar nova página\n");

printf("2. Voltar página anterior\n");

printf("3. Ver página atual\n");

printf("4. Mostrar histórico\n");

printf("0. Sair\n");

printf("Escolha uma opção: ");

scanf("%d", &opcao);

getchar(); // Limpa o '\n' deixado pelo scanf

switch (opcao) {

case 1:

printf("Digite a URL da nova página: ");

fgets(url, sizeof(url), stdin);

url[strcspn(url, "\n")] = '\0'; // Remove o \n do final

acessarPagina(&pilha, url);

break;

case 2:

voltarPagina(&pilha);

break;

case 3:

mostrarPaginaAtual(pilha);

break;

case 4:

mostrarHistorico(pilha);

break;

case 0:

printf("Saindo do navegador...\n");

break;

default:

printf("Opção inválida. Tente novamente.\n");

}

} while (opcao != 0);

// Libera memória restante (pilha completa)

while (pilha != NULL) {

voltarPagina(&pilha);

}

return 0;

}

**Filas – Exemplo 1**

#include <stdio.h>

#include <stdlib.h>

// Definição da estrutura do nó da fila

struct No {

int valor;

struct No \*prox;

};

// Função para enfileirar (enqueue): insere no fim da fila

void enqueue(struct No \*\*inicio, struct No \*\*fim, int valor) {

struct No \*novo = (struct No\*) malloc(sizeof(struct No)); // Aloca novo nó

novo->valor = valor; // Atribui valor ao novo nó

novo->prox = NULL; // Novo nó será o último, então prox = NULL

if (\*fim == NULL) {

// Se a fila está vazia, novo nó será o primeiro e o último

\*inicio = novo;

\*fim = novo;

} else {

(\*fim)->prox = novo; // Conecta o atual último ao novo nó

\*fim = novo; // Atualiza o fim da fila

}

}

// Função para desenfileirar (dequeue): remove do início da fila

int dequeue(struct No \*\*inicio, struct No \*\*fim) {

if (\*inicio == NULL) {

printf("Fila vazia!\n");

return -1;

}

struct No \*temp = \*inicio; // Guarda o nó a ser removido

int valor = temp->valor; // Armazena o valor para retorno

\*inicio = temp->prox; // Avança o início da fila

if (\*inicio == NULL) {

// Se a fila ficou vazia, zera também o fim

\*fim = NULL;

}

free(temp); // Libera a memória do nó

return valor;

}

// Função para consultar o início da fila (peek)

int peek(struct No \*inicio) {

if (inicio == NULL) {

printf("Fila vazia!\n");

return -1;

}

return inicio->valor; // Retorna o valor do início sem remover

}

// Função para imprimir a fila do início ao fim

void imprimirFila(struct No \*inicio) {

printf("Início da fila\n");

while (inicio != NULL) {

printf(" %d\n", inicio->valor);

inicio = inicio->prox;

}

printf("NULL (fim)\n");

}

// Função principal

int main() {

struct No \*inicio = NULL; // Ponteiro para o início da fila

struct No \*fim = NULL; // Ponteiro para o fim da fila

// Enfileira três elementos

enqueue(&inicio, &fim, 10);

enqueue(&inicio, &fim, 20);

enqueue(&inicio, &fim, 30);

// Mostra a fila atual

imprimirFila(inicio);

// Consulta o primeiro elemento

printf("Início da fila (peek): %d\n", peek(inicio));

// Remove um elemento da fila

int removido = dequeue(&inicio, &fim);

printf("Removido da fila (dequeue): %d\n", removido);

// Mostra a fila após remoção

imprimirFila(inicio);

return 0;

}

**Exemplo 2:**

#include <stdio.h>

#include <stdlib.h>

// Estrutura do paciente (nó da fila)

struct Paciente {

char nome[30]; // Nome (sem espaços)

int idade; // Idade

struct Paciente \*prox; // Próximo paciente na fila

};

// Inserir paciente no fim da fila (enqueue)

void enfileirar(struct Paciente \*\*inicio, struct Paciente \*\*fim) {

struct Paciente \*novo = (struct Paciente\*) malloc(sizeof(struct Paciente)); // Cria novo paciente

printf("Digite o nome do paciente (sem espaços): ");

scanf("%s", novo->nome); // Lê nome do paciente

printf("Digite a idade do paciente: ");

scanf("%d", &novo->idade); // Lê idade

novo->prox = NULL; // Como é o último da fila, não aponta para ninguém

if (\*inicio == NULL) {

// Se a fila estiver vazia, novo será o primeiro e o último

\*inicio = novo;

\*fim = novo;

} else {

// Se já houver pacientes, novo vai para o fim

(\*fim)->prox = novo;

\*fim = novo;

}

printf("Paciente inserido na fila com sucesso.\n");

}

// Atender paciente (remover do início da fila)

void atenderPaciente(struct Paciente \*\*inicio, struct Paciente \*\*fim) {

if (\*inicio == NULL) {

printf("Fila vazia! Nenhum paciente para atender.\n");

return;

}

struct Paciente \*temp = \*inicio; // Pega o paciente que será atendido

printf("Atendendo paciente: %s (idade %d)\n", temp->nome, temp->idade);

\*inicio = temp->prox; // Avança o início da fila para o próximo

if (\*inicio == NULL) {

// Se a fila ficou vazia após atender, fim também vira NULL

\*fim = NULL;

}

free(temp); // Libera a memória do paciente atendido

}

// Mostrar todos os pacientes da fila

void mostrarFila(struct Paciente \*inicio) {

if (inicio == NULL) {

printf("Fila vazia!\n");

return;

}

printf("\nFila de pacientes:\n");

while (inicio != NULL) {

printf("- %s (idade %d)\n", inicio->nome, inicio->idade);

inicio = inicio->prox; // Avança para o próximo paciente

}

}

// Função principal com menu

int main() {

struct Paciente \*inicio = NULL; // Ponteiro para o início da fila

struct Paciente \*fim = NULL; // Ponteiro para o fim da fila

int opcao;

do {

// Mostra o menu

printf("\n=== MENU - FILA DE PACIENTES ===\n");

printf("1. Inserir paciente na fila\n");

printf("2. Atender próximo paciente\n");

printf("3. Mostrar fila\n");

printf("0. Sair\n");

printf("Escolha uma opção: ");

scanf("%d", &opcao); // Lê a opção escolhida

// Executa a opção escolhida

if (opcao == 1) {

enfileirar(&inicio, &fim);

} else if (opcao == 2) {

atenderPaciente(&inicio, &fim);

} else if (opcao == 3) {

mostrarFila(inicio);

} else if (opcao == 0) {

printf("Encerrando o programa...\n");

} else {

printf("Opção inválida. Tente novamente.\n");

}

} while (opcao != 0); // Repete até o usuário escolher sair

// Libera a memória da fila (se ainda restar alguém)

while (inicio != NULL) {

atenderPaciente(&inicio, &fim);

}

return 0;

}