Plano de Estudos para Validação de Parênteses em C

Para resolver o exercício de validação de parênteses em C puro, você precisará dominar os seguintes tópicos:

1. Conceitos Fundamentais de C

- **Entrada e Saída (I/O):**
  - `scanf()` e `printf()` para leitura e escrita de dados formatados.
  - `gets()` ou `fgets()` para ler linhas de texto (expressões) que podem conter espaços.
- **Tipos de Dados:**
  - `char` para caracteres individuais.
  - `int` para contadores e o número de expressões (N).
- **Estruturas de Controle de Fluxo:**
  - `for` e `while` para iterar sobre as expressões e os caracteres dentro de cada expressão.
  - `if` e `else` para tomar decisões baseadas nas regras de validação dos parênteses.

2. Manipulação de Strings em C

- **Arrays de Caracteres:** Entender como strings são representadas em C como arrays de caracteres terminados por '\0'.
- **Funções de String (string.h):**
  - `strlen()` para obter o comprimento da string.
  - Acessar caracteres individuais da string usando `[]` (ex: `expressao[i]`).

3. Estruturas de Dados: Pilhas (Stacks)

Este é o conceito mais crucial para resolver este problema. Uma pilha é uma estrutura de dados LIFO (Last In, First Out - Último a Entrar, Primeiro a Sair).

- **Conceito de Pilha:** Entender como uma pilha funciona (operações de `push` e `pop`).
- **Implementação de Pilha em C:**
  - Usando um array: Criar um array de caracteres para armazenar os parênteses de abertura e um índice para controlar o topo da pilha.

- Operações `push` (adicionar um elemento ao topo) e `pop` (remover um elemento do topo).
- Verificar se a pilha está vazia (`isEmpty`) ou cheia (`isFull`).

4. Lógica de Validação de Parênteses

O algoritmo geral para validar parênteses usando uma pilha é o seguinte:

1. Inicialize uma pilha vazia.
2. Percorra a expressão caractere por caractere.
3. Se encontrar um parêntese de abertura `(` :
   - Adicione-o à pilha (`push`).
4. Se encontrar um parêntese de fechamento `)` :
   - Verifique se a pilha está vazia. Se estiver, a expressão é inválida (parêntese de fechamento sem correspondente de abertura).
   - Se a pilha não estiver vazia, remova o elemento do topo (`pop`).
5. Após percorrer toda a expressão:
   - Se a pilha estiver vazia, a expressão é válida (todos os parênteses foram corretamente fechados).
   - Se a pilha não estiver vazia, a expressão é inválida (parênteses de abertura não foram fechados).

5. Gerenciamento de Memória (Básico)

- Alocação estática para arrays (para a string da expressão e para a pilha).

Exemplo de Estrutura de Código (Pseudocódigo)

Plain Text

```
#include <stdio.h>
#include <string.h>
#include <stdbool.h>

// Definição e operações da pilha (push, pop, isEmpty)

int main() {
    int N;
    scanf("%d", &N);
    getchar(); // Consumir o newline após ler N
```

```
    for (int i = 0; i < N; i++) {
        char expressao[1001]; // +1 para o caractere nulo
        fgets(expressao, sizeof(expressao), stdin);
        expressao[strcspn(expressao, "\n")] = 0; // Remover o newline lido
por fgets

        // Lógica de validação usando a pilha
        bool valido = true;
        // ... (implementação da pilha e lógica de validação)

        if (valido) {
            printf("correct\n");
        } else {
            printf("incorrect\n");
        }
    }

    return 0;
}
```

Estudar e praticar esses tópicos, com foco especial na implementação e uso de pilhas, será fundamental para resolver o exercício. Comece com exemplos simples e gradualmente aumente a complexidade. Boa sorte!