Universidade Federal de Itajubá
Instituto de Matemática e Computação
Algoritmos e Estruturas de Dados II
CTCO-01
Vanessa Souza

Atividades de Fixação de Conteúdo

Assunto : TAD Pilha e TAD Fila

Questão 1: Structs e Operadores
Conceitos:
Os operadores . e -> são utilizados em C para acessar atributos de uma struct. O operador ponto é utilizado para acessar membros de uma estrutura diretamente quando você tem uma variável do tipo struct. Já o operador seta é utilizado para acessar membros de uma estrutura através de um ponteiro para a struct.

Quando você usa o typedef sem revelar a definição completa da struct, as funções que manipulam a struct devem usar ponteiros para operar sobre os dados. Isso é uma forma de encapsulamento, onde você esconde a implementação da struct em um arquivo .c e expõe apenas as funções para manipulá-la através de ponteiros.

Portanto, quando usamos TAD, a struct fica encapsulada no arquivo .c e o programa usuário tem acesso apenas ao arquivo .h. Ou seja, na main, não há acesso direto à struct, apenas ao seu typedef. Por isso, precisamos utilizar ponteiro para acessar essa estrutura. E, ao usar ponteiro, a sintaxe correta é seta e não ponto.

Atividade:
Considere o arquivo Lista_3_Pilha_Fila_Anexo.c Estude o arquivo (QUESTÃO 1) e complete o código nos locais que são solicitados. Posteriormente, execute o programa. Qual a saída do programa?

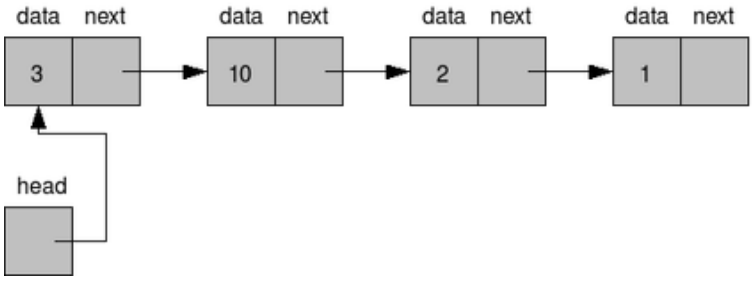
O código abaixo funciona? Explique.

```
printf("Nome: %s\n", (*f2).nome);
printf("Idade: %d\n", (*f2).idade);
printf("Salario: %.2f\n", (*f2).salario);
```

Questão 2: Navegando em Estruturas Dinâmicas (Encadeadas)
Conceitos:
Como vimos estruturas encadeadas apresentam como principal característica um elemento nó que guarda os dados e um ponteiro para o próximo nó. O ponteiro é um endereço para outro nó na memória RAM. Um nó é acessado a partir de outro. O início da navegação são os marcadores de início da estrutura.

Considerando a ilustração abaixo, a navegação começa em *head*, que acessa o primeiro nó da estrutura que, no caso, é uma lista. Portanto, *lista->head* é um ponteiro para o nó 3 e, na prática, acessa o nó 3.

Portanto, se fizermos lista->head->next, acessaremos o nó 10.
Se fizermos lista->head-> next -> next, acessaremos o nó 2.
De forma análoga, se fizermos lista->head-> next -> next->next, acessamos o nó 1
Finalmente, se fizermos lista->head-> next -> next->next->next, teremos o valor NULL.



Atividade:
Considere o arquivo Lista_3_Pilha_Fila_Anexo.c Estude o arquivo (QUESTÃO 2). Implemente as funções criaPilha (uma pilha cujos nós armazenam caracteres), push e a percorrePilha. Sabemos que não é permitido percorrer uma pilha. Esse exercício é apenas para fixar o conceito acima estabelecido. Na função percorrePilha, utilizar apenas o topo da pilha para imprimir todos os três elementos empilhados. A saída deve ser:
c
b
a

Questão 3: Verificação de Expressão Matemática Balanceada
Você recebeu uma *string* que contém uma expressão matemática, a qual inclui parênteses (), colchetes [] e chaves {}. Sua tarefa é escrever um programa que verifique se todos os símbolos de abertura e fechamento estão **balanceados**.
Regras:

1. Para cada parêntese de abertura (deve haver um correspondente parêntese de fechamento), e o mesmo vale para colchetes [] e chaves {}.
2. A ordem correta deve ser mantida. Por exemplo, a expressão ([ ]) está balanceada, mas a expressão ( ]) não está.

Exemplos:

- A expressão [{(2+3)*5} + 3] está **balanceada**.
- A expressão [(3*2) + {4*5)] está **desbalanceada**.

Entrada:

- Uma única string representando a expressão matemática (a string pode conter operadores e números e deve ter tamanho máximo de 30 caracteres).

Saída:

- Retorne "BALANCEADA" se a expressão estiver balanceada.
- Retorne "DESBALANCEADA" se a expressão estiver desbalanceada.

Exemplo de funcionamento do programa:

```
Digite a expressao matematica: [{(2+3)*5} + 3]
BALANCEADA
```

```
Digite a expressao matematica: [(3*2) + {4*5)]
DESBALANCEADA
```

Questão 4: Sistema de Atendimento em um Restaurante

Enunciado:
Você foi contratado para criar o sistema de gerenciamento de atendimento de um restaurante. O restaurante tem uma fila de espera para os clientes que chegam. Os clientes são atendidos em ordem de chegada, e o sistema deve gerenciar essa fila de maneira eficiente.
Cada cliente tem um nome e o número de pessoas que estão com ele. O sistema precisa controlar a ordem de chegada, atender os clientes na sequência correta e liberar os clientes da fila após o atendimento.
Funcionalidades:

1. **Adicionar um cliente à fila**:
    - Quando um cliente chega, seu nome e o número de pessoas que o acompanham são adicionados à fila de espera. O cliente é informado sobre a quantidade de pessoas a sua frente.
2. **Atender um cliente**:
    - O cliente no início da fila é removido, e seu nome e o número de pessoas são mostrados para o atendente.
3. **Exibir o tamanho da fila de espera**:
    - O sistema deve exibir quantos clientes estão na fila de espera.

Detalhes Técnicos:

1. Crie uma **estrutura de dados Fila**, onde cada nó da fila deve conter:
    - O nome do cliente (máximo de 50 caracteres).
    - O número de pessoas que o cliente trouxe.
    - Um ponteiro para o próximo cliente.
2. Implemente as seguintes funções:
    - **criaFila():** Cria uma fila que armazena o total de clientes em espera.
    - **adicionarCliente(Fila *f, char nome[], int numPessoas)**: Adiciona um novo cliente à fila. Retorna 1 caso o cliente seja inserido com sucesso à fila e 0 caso contrário.
    - **atenderCliente(Fila *f)**: Retira um cliente da fila. Retorna 1 caso o cliente seja atendido com sucesso e 0 caso contrário.
    - **TamanhoFila(Fila *f)**: Retorna o tamanho da fila de espera.
    - **ClienteTopo(Fila *f):** Retorna o nome do cliente que está no início da fila. Se a fila estiver vazia, retorna NULL
    - **TotalClientesTopo (Fila *f):** Retorna o total de clientes no início da fila. Se a fila estiver vazia, retorna 0.

- **desalocaFila(Fila *f):** desaloca a fila. Caso haja clientes na fila, os mesmos deverão ser retirados antes da desalocação da fila.

3. Implemente um programa usuário (main) que faça o seguinte fluxo:
   - Cria a fila
   - Insira 5 clientes
   - Remova 1 cliente
   - Insira 1 cliente
   - Remova 2 clientes
   - Verifica o tamanho da fila
   - Desaloca fila

Regras:

- Se a fila estiver vazia ao tentar atender um cliente, o sistema deve exibir uma mensagem adequada.
- O sistema deve garantir que os clientes sejam atendidos na ordem em que chegaram.
- Para cada cliente inserido, informar a quantidade de clientes na fila antes dele.
- Depois de cada cliente atendido, mostrar na tela o nome do próximo cliente e para quantas pessoas deverá ser a mesa.

Abaixo um exemplo de execução do programa:

```
Há 0 pessoas na frente do cliente 1
Há 3 pessoas na frente do cliente 2
Há 8 pessoas na frente do cliente 3
Há 10 pessoas na frente do cliente 4
Há 11 pessoas na frente do cliente 5

 Próximo Cliente a ser atendido
Nome: José
Mesa para 5 pessoas

Há 11 pessoas na frente do cliente 6
Há 15 pessoas na frente do cliente 7

 Próximo Cliente a ser atendido
Nome: Eva
Mesa para 2 pessoas


 Próximo Cliente a ser atendido
Nome: Josefina
Mesa para 1 pessoas

Ainda há 10 pessoas na fila
```

Questão 5: Altere o programa da Questão 4 para que o cliente saiba em qual posição da fila ele entrou.

Questão 6: Gerenciamento de Pool de Impressoras

Enunciado:

Uma empresa possui duas impressoras para impressão de documentos. Cada impressora tem sua própria fila de documentos a serem impressos. A cada novo documento que chega, ele é colocado na fila da impressora que estiver com menos documentos no momento. Se ambas as filas tiverem o mesmo número de documentos, o documento é enviado para a impressora 1.

Cada documento tem um tempo de impressão associado, e a ordem de impressão deve seguir o conceito de fila (FIFO - First In, First Out).

Seu trabalho é implementar esse sistema de gerenciamento de impressoras com as funções listadas no arquivo Lista_3_Pilha_Fila_Anexo.c (QUESTÃO 6).

Cada documento possui:

- Um identificador (número único).
- Um tempo de impressão (em segundos).

O programa deve:

- Receber uma quantidade indefinida de documentos e inseri-los na fila da impressora que estiver com menos documentos.
- As impressões devem ocorrer de forma intercala com as inserções de novos documentos.

Você deve criar o programa usuário.