BoltFinance - Seu App Financeiro

Bloco 1: Inclusões e Definições

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

#define MAX_DESPESAS 100
```

- **Inclusões:** Importa bibliotecas padrão para entrada/saída, alocação de memória e manipulação de strings.
- **Definição:** Estabelece uma macro `MAX_DESPESAS` com o valor 100 para limitar o número máximo de despesas.

Bloco 2: Estrutura de Dados

```
struct Despesa {
    double valor;
    char data[11];
    char categoria[50];
};
```

- **Estrutura de Dados:** Define uma estrutura chamada `Despesa` com três campos: `valor` (double), `data` (string de 11 caracteres) e `categoria` (string de 50 caracteres).

Bloco 3: Protótipos de Funções

```
void exibirMenu();
void limparBuffer();
```

```
void consultarSaldo(double *saldo, struct Despesa *despesas, int numDespesas);
void registrarDespesa(struct Despesa *despesa, int *numDespesas);
void adicionarSaldo(double *saldo);
void listarDespesas(struct Despesa *despesas, int numDespesas);
void exibirCabecalho(const char *titulo);
```

- **Protótipos:** Declaração antecipada das funções para que o compilador as reconheça antes de serem definidas.

```
void consultarSaldo(double *saldo, struct Despesa *despesas, int numDespesas);
```

Explicação:

- `void`: Indica que a função não retorna nenhum valor.
- `consultarSaldo`: Nome da função.
- `(double *saldo, struct Despesa *despesas, int numDespesas)`: Parâmetros da função.
  - `double *saldo`: Um ponteiro para uma variável do tipo double, representando o saldo.
  - `struct Despesa *despesas`: Um ponteiro para uma estrutura `Despesa`, representando um array de despesas.
  - `int numDespesas`: Um inteiro representando o número de despesas no array.

Essa linha declara um protótipo de função chamada `consultarSaldo` que não retorna nenhum valor (`void`). Ela recebe três parâmetros: um ponteiro para `double` chamado `saldo`, um ponteiro para uma estrutura `Despesa` chamado `despesas`, e um inteiro chamado `numDespesas`.

Essa função é projetada para consultar o saldo, exibir as despesas registradas e atualizar o saldo subtraindo o valor das despesas. O uso de ponteiros permite que a função modifique diretamente as variáveis fora da função, garantindo uma atualização eficaz do saldo e das despesas.

Bloco 4: Função Principal `main()`

```
int main() {
    // Declaração de variáveis
    double saldo = 1000.0;
    struct Despesa despesas[MAX_DESPESAS];
    int numDespesas = 0;
    int opcao;

    // Loop principal do menu
    do {
        // Exibe o menu
        exibirMenu();

        // Recebe a opção do usuário
        while (scanf("%d", &opcao) != 1) {
            limparBuffer();
            printf("Opção inválida. Por favor, escolha uma opção válida: ");
        }

        // Executa a opção escolhida
        switch (opcao) {
            case 1:
                consultarSaldo(&saldo, despesas, numDespesas);
                break;
            case 2:
                registrarDespesa(despesas, &numDespesas);
                break;
            case 3:
                adicionarSaldo(&saldo);
                break;
            case 4:
                listarDespesas(despesas, numDespesas);
                break;
            case 5:
                printf("\\nSaindo do aplicativo. Até logo!\\n");
                break;
            default:
                printf("\\nOpção inválida. Por favor, escolha uma opção válida.\\n");
        }

    } while (opcao != 5);

    return 0;
}
```

- **Variáveis:** Declaração de variáveis para saldo, despesas e opção do usuário.
- **Loop Principal:** `do-while` para exibir o menu e processar a escolha do usuário.

- **Switch:** Estrutura de controle para executar a função correspondente à opção escolhida.

Bloco 5: Funções do Menu

```
void exibirMenu() {
    exibirCabecalho("BoltFinance");
    printf("1. Consultar saldo\\n");
    printf("2. Registrar despesas\\n");
    printf("3. Adicionar saldo\\n");
    printf("4. Listar despesas\\n");
    printf("5. Sair\\n");
    printf("Escolha uma opção: ");
}

void limparBuffer() {
    int c;
    while ((c = getchar()) != '\\n' && c != EOF);
}
```

- `exibirMenu()`**:** Mostra o menu principal chamando `exibirCabecalho()` e exibindo opções numeradas.
- `limparBuffer()`**:** Limpa o buffer de entrada para evitar problemas com entradas inválidas.

Bloco 6: Funções de Funcionalidades

```
void consultarSaldo(double *saldo, struct Despesa *despesas, int numDespesas) {
    // Implementação da funcionalidade de consultar saldo
}

void registrarDespesa(struct Despesa *despesa, int *numDespesas) {
    // Implementação da funcionalidade de registrar despesa
}

void adicionarSaldo(double *saldo) {
    // Implementação da funcionalidade de adicionar saldo
}

void listarDespesas(struct Despesa *despesas, int numDespesas) {
    // Implementação da funcionalidade de listar despesas
}

void exibirCabecalho(const char *titulo) {
```

```
    // Exibe um cabeçalho formatado com o título fornecido
}
```

- **Implementações de Funcionalidades:** Cada função realiza uma funcionalidade específica, como consultar saldo, registrar despesa, adicionar saldo, listar despesas e exibir cabeçalho.

Conclusão:

Este código implementa um aplicativo financeiro simples em C, fornecendo um menu interativo para o usuário realizar operações como consultar saldo, registrar despesas, adicionar saldo e listar despesas. A estrutura modular e as funções claras facilitam a manutenção e compreensão do código.