Documentação do Sistema de Reserva de Assentos

Este programa em C simula um sistema de reserva de assentos de avião, onde o usuário pode escolher a fileira e a poltrona para a reserva. Ele também valida as opções escolhidas com base no tipo de passagem e se o assento já está ocupado ou não. A seguir, explicamos o funcionamento detalhado de cada parte do código.

Declaração de Variáveis

```
char reserva[10][6];
char continuar;
int fileira = -1, assento = -1;
char poltrona = ' ';
char tipo_passagem;
```

Como vimos em aula, essas variáveis são declaradas no início do código. O array `reserva` é uma matriz de 10 fileiras e 6 colunas que representa os assentos de um avião. Os outros são variáveis auxiliares que armazenam a fileira, o assento, a poltrona escolhida e o tipo de passagem.

Interface de Escolha do Tipo de Passagem

```
do {
    printf("\nEscolha o tipo de passagem (E - Econômica, X
- Executiva): ");
    scanf(" %c", &tipo_passagem);
    if (tipo_passagem != 'E' && tipo_passagem != 'X')
        printf("\nTipo de passagem inválido. Tente novament
e.\n");
} while (tipo_passagem != 'E' && tipo_passagem != 'X');
```

Este bloco de código utiliza um laço `do-while`, como vimos em aula, para garantir que o usuário escolha um tipo de passagem válido. Ele só sai do loop

quando o usuário digita 'E' para Econômica ou 'X' para Executiva.

Validação de Fileira e Poltrona

```
while (true) {
    printf("\nDigite a fileira (1-10): ");
    scanf("%d", &fileira);
    printf("\nDigite a poltrona [A][B][C][D][E][F]: ");
    scanf(" %c", &poltrona);
```

Aqui temos uma verificação contínua que solicita ao usuário que insira a fileira (entre 1 e 10) e a poltrona (entre A e F). Essa validação impede que o usuário selecione assentos fora do intervalo.

Conversão da Poltrona para Índices da Matriz

```
switch (poltrona) {
    case 'A': case 'a': assento = 0; break;
    case 'B': case 'b': assento = 1; break;
    case 'C': case 'c': assento = 2; break;
    case 'D': case 'd': assento = 3; break;
    case 'E': case 'e': assento = 4; break;
    case 'F': case 'f': assento = 5; break;
    default:
        printf("\nPoltrona inválida\n");
        continue;
}
```

Como discutimos em aula, o uso de `switch-case` é uma maneira eficiente de mapear as opções de poltronas (A-F) para os índices correspondentes no array `reserva`.

Verificação de Assento Disponível

```
if (reserva[fileira - 1][assento] == 'x') {
    printf("\nEsse assento já está reservado. Por favor, es
colha outro.\n");
```

```
    continue;
}
```

Neste trecho, o código verifica se o assento escolhido já está reservado, usando o valor `'x'` como indicador de reserva. Se estiver ocupado, o usuário é solicitado a escolher outro.

Restrições para a Passagem Econômica

```
if (tipo_passagem == 'E' && (assento == 0 || assento == 5))
{
    printf("\nPara passagem Econômica, as poltronas A e F n
ão estão disponíveis.\n");
    continue;
}
```

Aqui o programa impõe uma restrição para passagens econômicas, onde as poltronas A e F (assentos nas extremidades) não podem ser reservadas. Isso demonstra como a lógica condicional é usada para aplicar regras específicas de negócios.

Exibição do Mapa de Assentos

```
printf("\n\t\t[A] [B] [C]\t[D] [E] [F]\n");
for (int x = 0; x < 10; x++) {
    printf("\n\t%02d\t", x + 1);
    for (int y = 0; y < 6; y++) {
        if (reserva[x][y] == 'x') {
            printf("[🔴] "); // Assento ocupado
        } else {
            printf("[🟢] "); // Assento livre
        }
        if (y == 2) {
            printf("\t");
        }
    }
}
```

Este bloco exibe graficamente o estado atual dos assentos no avião. O uso de emojis, como o círculo vermelho `[🔴]` para assentos ocupados e o círculo verde `[🟢]` para assentos livres, facilita a visualização da disponibilidade de assentos, como discutimos em relação à melhoria da experiência do usuário nas nossas aulas.

Encerramento do Sistema de Reservas

```
do {
    printf("\nDeseja realizar outra reserva? (s/n): ");
    scanf(" %c", &continuar);
} while (continuar == 's' || continuar == 'S');
```

No final, o programa pergunta se o usuário deseja fazer outra reserva e repete o processo caso a resposta seja afirmativa. Esta parte do código é importante para permitir que o usuário continue interagindo sem precisar reiniciar o sistema.

Essa documentação descreve cada etapa do código e como ela se relaciona com os conceitos que vimos durante as aulas, incluindo laços de repetição, validação de entrada, estruturas de controle e manipulação de arrays.

Se tiver alguma dúvida ou sugestão, fique à vontade para entrar em contato.