

Décimo Primeiro Laboratório de ECOP11

UNIFEI - Universidade Federal de Itajubá

Prof. Gabriel Cirac Mendes Souza (gabrielcirac@unifei.edu.br)

Assunto: Estruturas Heterogêneas

Vamos escrever um algoritmo que cria um baralho de cartas para o jogo “**Truco**” - com todos os naipes e valores deste tipo de jogo -, embaralha as cartas, as distribui aleatoriamente para dois jogadores (3 cartas para cada), e vira uma sobre a mesa. Para isso, iremos criar uma estrutura `carta`, e um tipo `carta_t` definido a partir desta estrutura. Siga os passos:

- 1) (20 pontos) Crie uma **struct** “`carta`” que possui dois valores: um inteiro representando seu valor e um caractere representando seu naipe. Na tabela ASCII, os valores dos naipes são respectivamente:

Copas	♥	3
Ouros	♦	4
Paus	♣	5
Espadas	♠	6

Portanto, basta que você os atribua com os valores inteiros 3, 4, 5 e 6 e os imprima como caracteres para que seu programa mostre os naipes corretamente.

- 2) (10 pontos) Utilizando a diretiva **typedef**, crie um tipo de dados a partir da struct `carta`, chamado “`carta_t`”.
- 3) (10 pontos) Crie, na função `main`, um vetor de “`carta_t`” de 40 posições, que representa o baralho completo de truco.

Observação: O baralho completo possui 52 cartas. No entanto, no truco, desprezamos três valores em prol da utilização do valete (valor 8), dama (valor 9), e rei (valor 10), resultando em um baralho de 40 cartas. Não utilizaremos nenhum coringa.

- 4) (25 pontos) Crie uma função para inicializar o baralho. Ela será responsável por dar valor a cada uma das quarenta cartas. Lembre-se que as cartas possuirão valores entre 1 (A, Ás) e 10 (K, Rei), e naipes entre 3 (copas) e 6 (espadas). Percorra as 40 posições do array e inicialize cada carta com seu valor apropriado.
- 5) (25 pontos) Crie uma função para embaralhar as cartas. Ela receberá um vetor de `carta_t` e fará 50 trocas de cartas. A função consiste em um laço que se repete por 50 vezes (for). Dentro dele, sorteamos duas cartas através da função `rand()`. A seguir, trocamos as duas cartas de posição dentro do baralho, utilizando uma `carta_t` auxiliar, como fizemos nos métodos de ordenação.

6) (10 pontos) Copie a função main abaixo e verifique se seu programa está correto:

```
// prototipo de funções
void inicializa_baralho(carta_t[]);
void embaralha(carta_t[]);

// Função Principal
int main()
{
    // Baralho com 40 cartas (A to K, copas to espadas)
    carta_t baralho[40];

    // Dá valores às cartas
    inicializa_baralho(baralho);

    // Embaralha
    srand(time(NULL));
    embaralha(baralho);

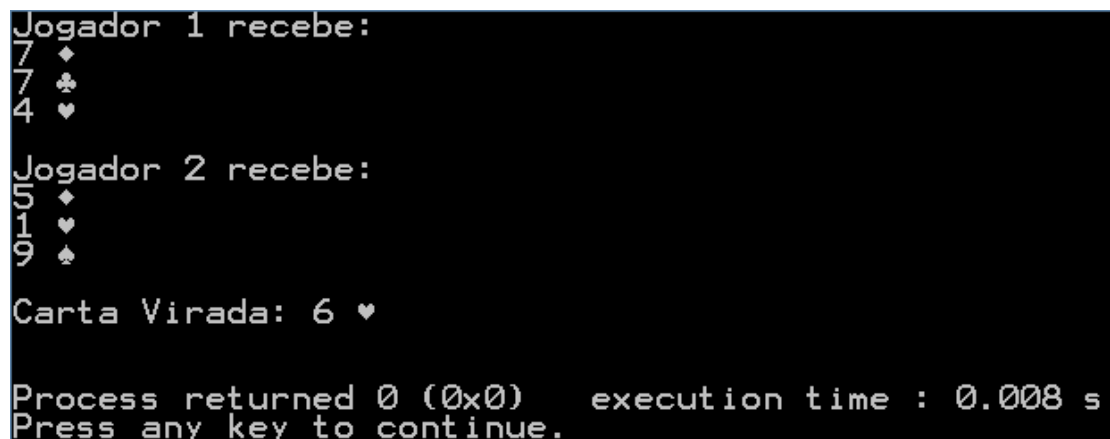
    // Distribuindo cartas
    printf("Jogador 1 recebe: \n");
    printf("%d %c \n", baralho[0].valor, baralho[0].naipe);
    printf("%d %c \n", baralho[1].valor, baralho[1].naipe);
    printf("%d %c \n\n", baralho[2].valor, baralho[2].naipe);

    printf("Jogador 2 recebe: \n");
    printf("%d %c \n", baralho[3].valor, baralho[3].naipe);
    printf("%d %c \n", baralho[4].valor, baralho[4].naipe);
    printf("%d %c \n\n", baralho[5].valor, baralho[5].naipe);

    printf("Carta Virada: %d %c \n\n", baralho[6].valor,
    baralho[6].naipe);

    return 0;
}
```

Veja o exemplo de saída:



```
Jogador 1 recebe:
7 ♦
7 ♣
4 ♥

Jogador 2 recebe:
5 ♦
1 ♥
9 ♠

Carta Virada: 6 ♥

Process returned 0 (0x0)   execution time : 0.008 s
Press any key to continue.
```