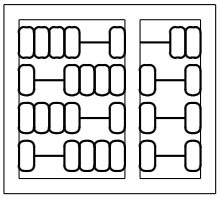

Laboratório 02 - *High Card*

Solução comentada

MC102 - Algoritmos e Programação de Computadores
Instituto de Computação - UNICAMP
Prof.: Hélio Pedrini

Solução por
Alexandre Medeiros – alexandre.medeiros@students.ic.unicamp.br

Problema

Considere um jogo de baralho, no qual 2 jogadores recebem uma carta e aquele com a carta de maior valor ganha. O baralho considerado possui 52 cartas distribuídas em quatro grupos denominados naipes. Cada naipe possui 13 cartas, sendo elas: um ás (A), todos os números de 2 a 10, e três figuras: *jack* (J), *queen* (Q) e *king* (K). Os naipes são *spades* (S), *hearts* (H), *diamonds* (D) e *clubs* (C). A figura 1 mostra um baralho com seus naipes e valores.

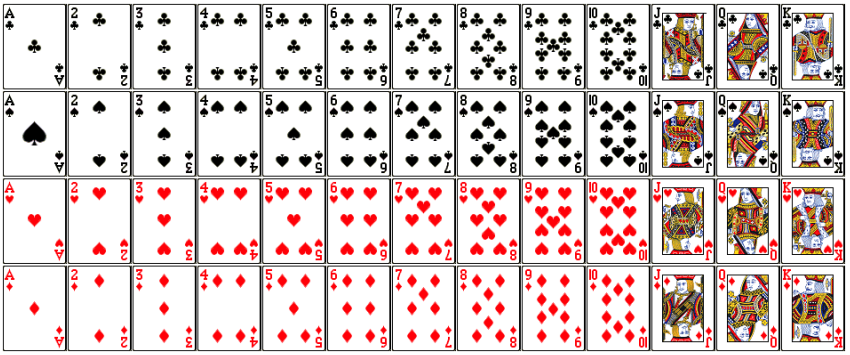

Figura 1: Baralho

Para determinar a carta vencedora, deve-se considerar a seguinte ordem de valores:

$$2 < 3 < 4 < 5 < 6 < 7 < 8 < 9 < 10 < J < Q < K < A \tag{1}$$

No caso de empate, deve-se avaliar o naipe assumindo a seguinte ordem:

$$S < H < D < C \tag{2}$$

Crie um programa que recebe como entrada duas cartas e retorna a carta vencedora.

Entrada

A entrada é composta de duas linhas, cada linha possui uma carta com uma representação do valor e uma representação do naipe. Os números de 2 a 9 representam as cartas com seu respectivo valor. Os outros valores e os naipes são mostrados nas Tabelas 1 e 2.

Solução

Em comparação ao primeiro laboratório, este problema não introduz a necessidade de estruturas novas, porém traz conceitos importantes. A tarefa necessita apenas de comandos condicionais, como *if* e *case*. Note que tudo que pode ser feito com um *if*, também pode ser feito com um *case*, porém um deles

Valor	Representação
10	0
Ás	A
Jack	J
Queen	Q
King	K

Tabela 1: Representação de cartas

Naipe	Representação
Clubs	C
Diamonds	D
Hearts	H
Spades	S

Tabela 2: Representação de naipes

pode ser mais adequado que o outro dependendo da situação, especialmente quando tornam o código mais simples e legível.

Para resolver o problema, primeiro precisamos ler as entradas e nesse ponto já encontramos nossas primeiras dificuldades. Note que todos os valores de cartas são representados ou por um número de um dígito ou por uma letra, com isso pode-se notar uma "dica" de que devemos utilizar o tipo de variável *char* para armazenar estes dados, pois assim conseguiríamos lê-los tanto quando estes são letras ou números.

Ler os valores das cartas como caracteres nos traz um problema, não podemos usá-los para fazer as comparações diretamente. Poderíamos fazer comparações do tipo "se a carta um é um Ás e a carta dois é um *King* então a carta um ganha", porém para cobrir todos os casos de entrada, acabariamos fazendo um programa enorme que não é nem um pouco eficiente. Mas qual seria uma boa alternativa?

É simples, não precisamos necessariamente utilizar os valores lidos para fazer as comparações, podemos "convertê-los" para outros valores, mais fáceis de trabalhar internamente, para isso, podemos utilizar duas variáveis auxiliares, que devem armazenar nosso valor atribuído de cada carta.

Agora precisamos nos decidir como que serão esses valores intermediários que usaremos. Quais valores seriam mais práticos para nós? Valores que possam ser usados para a comparação, de tal forma que se uma carta é maior que a outra, o seu valor atribuído seja maior que o da outra carta. Uma possível escolha de valores, está exemplificada na Tabela 3.

Valor da carta	Valor atribuído
2 até 10	O valor da própria carta
Jack	11
Queen	12
King	13
Ás	14

Tabela 3: Valor intermediário atribuído às cartas

Escolhidos os nossos valores intermediários, agora precisamos atribuí-los às nossas variáveis auxiliares, agora vem um detalhe muito importante: as entradas estão em *char*, ou seja, estão na representação da Tabela ASCII[1], que nada mais é do que uma associação de um caractere para cada número de 0 até 127. Observe que dentro do computador os caracteres são representados apenas com números, apenas quando é necessário imprimi-los que o computador procura o desenho associado àquela letra.

Algo importante de se notar na Tabela ASCII, é que os números (de 0 até 9) estão distribuídos sequencialmente e em ordem crescente, mas os números atribuídos aos caracteres numéricos não são os números que esses caracteres representam, ou seja, o caractere '6' não é representado no computador como o número 6, mas sim o número 54.

Agora que sabemos os detalhes de como o computador representa os caracteres internamente, podemos nos aproveitar disso para determinar o nosso valor atribuído de cada carta. Note que das cartas 2 até a 9, para obtermos o valor que queremos, tudo que precisamos fazer é subtrair um valor do número usado para representar o seu dígito, no caso o número 48, que representa o caractere '0'. Algo muito útil em `C` é a possibilidade de não precisar saber que o caractere '0' é o número 48, basta se lembrar que os dígitos numéricos estão em ordem e são consecutivos. No seu código, você pode colocar "variável - '0' " que o compilador irá interpretar isso como subtraia o número que representa o caractere '0' na Tabela ASCII da variável.

[1] `www.asciitable.com`

Programa 1: Transforma o dígito lido em um valor facilmente comparável

```
    switch(c0_dig) {
        case 'A':   /*  A  */
            c0_val = 14;
            break;
        case 'K':   /*  K  */
            c0_val = 13;
            break;
        case 'Q':   /*  Q  */
            c0_val = 12;
            break;
        case 'J':   /*  J  */
            c0_val = 11;
            break;
        case '0':   /*  10 */
            c0_val = 10;
            break;
        default:    /* 2-9 */
            c0_val = c0_dig - '0';
    }
```

Note que o *switch* no trecho de Programa 1 torna o código bem simples e legível, onde você atribui um valor para cada caso do dígito da carta.

Com o valor de cada carta bem determinado, só nos resta compará-las. Você pode estar se peguntando: "Mas e o caso das cartas terem o mesmo valor? Não precisamos verificar os naipes?". Olhe bem a ordem de precedência dos naipes, repare que eles seguem uma ordem onde o naipe mais forte é o que seu símbolo vem primeiro na ordem alfabética e como vocês já sabem, as letras do alfabeto estão em ordem na tabela ASCII, logo, o caractere 'C' será menor que o caractere 'D' e assim por diante, então podemos usar o fato de que o naipe cuja letra é menor, é na verdade o maior naipe! Assim, temos nosso programa pronto com a verificação do *if* no trecho de Programa 2.

Programa 2: Comparação das cartas

```
    /* Faz comparacao do valor das cartas.  */
    if (c0_val > c1_val || (c0_val == c1_val && c0_suit < c1_suit))
        printf("%c␣%c\n", c0_dig, c0_suit);
    else
        printf("%c␣%c\n", c1_dig, c1_suit);
```