Bingo e Gerador de Cartelas Laboratório

David Sena *

1 Instruções Gerais

O objetivo desse laboratório é fazer um código que simule um bingo e um gerador de cartelas. Esse lab é o complemento do trabalho do Bingo.

1.1 Alimentando o Globo do bingo

Vamos precisar colocar bolas dentro do globo. Podemos utilizar um vector¡int¿ em C++ para guardar nossas bolas. Ou uma lista do Python. Ou um vetor de int mesmo do C. Se fizer em C, você vai ter que lembrar de passar o tamanho do vetor em todas as passagens de parametro.

Primeiro objetivo é colocar todas as bolas no vector.

```
Var _globo = vetor vazio;
Para cada _num de 1 ate 75 Faca:
    insira _num no _globo
```

Agora vem a parte divertida. Como sorteamos e retiramos um número do globo? Veja que cada posição do vetor contém um número que é válido. Precisamos escolher aleatoriamente uma posição no vetor. Para isso vamos utilizar a função `rand()`. O código `int pos = rand()% globo.size();` nos gera um número entre 0 e o tamanho do globo. Precisamos então RETIRAR essa bola do globo.

*sena.ufc@gmail.com

2 Retirando uma bola do Globo

Num vetor, não é interessante APAGAR um elemento no meio, pois isso implica em "puxar pra frente"todos os elementos que estão depois da posição apagada pra preencher o buraco que ficou.

Como a ordem dos elementos no nosso vetor globo não é importante, fica mais interessante apenas pegar o último elemento e trazer pra onde ficou o buraco. Depois apenas diminuímos o tamanho do vetor.

Que tal fazermos uma função que recebe um vetor, sorteia um elemento, remove o elemento do vetor e o retorna. Vamos lá:

```
int pegar_bola(_vetor):
    Var _posicao = sorteie posicao valida no vetor
    Var _valor = vetor[_posicao]
    copie o elemento do fim para _vetor[_posicao]
    remova o elemento do fim
    Retorne _valor
Fim
```

O interessante é que toda vez que chamamos nossa função, o vetor diminui de 1 no tamanho e um valor válido é retornado.

2.1 Guardando nossas bolas na mesa

Já temos o nosso globo. Precisamos agora de uma mesa pra guardar as bolas sorteadas. Fica fácil pensar na mesa como um outro vetor. Basta que todas as bolas que removemos do globo sejam colocadas na mesa. Como fazemos então uma impressão bonita como aquela do exemplo abaixo? Olhe que faltam as bolas 4, 36, 41, 44, etc. Essas bolas já foram sorteadas.

```
Globo:
 1  2  3 __  5  6  7  8  9 10 11 12 13 14 15
16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
31 32 33 34 35 __ 37 38 39 40 __ 42 43 __ __
46 47 48 49 50 51 52 53 54 55 __ 57 __ 59 60
61 62 63 __ 65 66 67 68 69 70 71 72 73 74 75
Escolha 1 para pedir bola e 0 para sair
```

Podemos fazer um laço de 1 até 75, onde a cada 15 elementos nós damos uma quebra de linha. Se o elemento existir no vetor, nos o imprimimos. Se

não existir, nós imprimimos um __. Para saber se o elemento existe no vetor, precisaremos percorrer o vetor todo procurando o elemento. Sabe de uma coisa, é mais prático fazer uma função pra isso. Que tal?

```
bool existe(_elemento, _vetor):
    Para cada _valor no _vetor:
        Se _valor eh igual a _elemento Entao:
            Retorne true
        Fim
    Retorne false
Fim
```

Se nosso código chegou ao fim do laço é porque o elemento não foi encontrado. Se não foi encontrado, então retorne false. Lembre de adaptar o psudocódigo à sua linguagem, ok?!

Estamos prontos para uma impressao bonitinha.

```
void imprime_formatado(_vetor):
    Para _num de 1 ate 75:
        Se existe(_num, _vetor):
            imprime _num
        Senao:
            imprime __
        Se _num eh multiplo de 15:
            imprime quebra de linha
```

Morreu Maria Preá! Você vai ter que aprender na sua linguagem como fixar a quantidade de dígitos na hora da impressão. Você pode olhar nos seguintes links: C LINK, Python LINK

Se tudo der certo você pode agora implementar um código com uma saída parecida com a seguinte:

```
Iniciando Bingo:

globo:
 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
31 32 33 34 35 36 37 38 39 40 41 42 43 44 45
46 47 48 49 50 51 52 53 54 55 56 57 58 59 60
61 62 63 64 65 66 67 68 69 70 71 72 73 74 75
Escolha 1 para pedir bola e 0 para sair
```

```
>> 1

Numero sorteado 57

Globo:
 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
31 32 33 34 35 36 37 38 39 40 41 42 43 44 45
46 47 48 49 50 51 52 53 54 55 56 __ 58 59 60
61 62 63 64 65 66 67 68 69 70 71 72 73 74 75

Mesa:
-- -- -- -- -- -- -- -- -- -- -- -- -- -- --
-- -- -- -- -- -- -- -- -- -- -- -- -- -- --
-- -- -- -- -- -- -- -- -- -- -- -- -- -- --
-- -- -- -- -- -- -- -- -- -- -- 57 -- -- --
-- -- -- -- -- -- -- -- -- -- -- -- -- -- --

Escolha 1 para pedir bola e 0 para sair
>> 1

Numero sorteado 58

globo:
 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
31 32 33 34 35 36 37 38 39 40 41 42 43 44 45
46 47 48 49 50 51 52 53 54 55 56 __ __ 59 60
61 62 63 64 65 66 67 68 69 70 71 72 73 74 75

Mesa:
-- -- -- -- -- -- -- -- -- -- -- -- -- -- --
-- -- -- -- -- -- -- -- -- -- -- -- -- -- --
-- -- -- -- -- -- -- -- -- -- -- -- -- -- --
-- -- -- -- -- -- -- -- -- -- -- 57 58 -- --
-- -- -- -- -- -- -- -- -- -- -- -- -- -- --
Escolha 1 para pedir bola e 0 para sair
>> 0
```

```
Obrigado e volte sempre.
```

3 Gerando a cartela

A segunda parte da implementação é a criação de um gerador de cartelas. A regra aqui é que cada letra contem bolas dentro de uma faixa.

- B - 5 bolas entre 1 e 15
- I - 5 bolas entre 16 e 30
- N - 4 bolas entre 31 e 45
- G - 5 bolas entre 46 e 60
- O - 5 bolas entre 61 e 75

Os números normalmente não são ordenados dentro da letra. A seguir o exemplo de uma cartela. Olhe que o N possui só 4 bolas. A bola do meio não existe.

```
Cartela 1
B  I  N  G  O
12 18 31 58 63
 3 19 43 54 67
 1 26 ## 59 71
10 23 45 50 68
14 21 38 60 65
```

Utilizando os conhecimento do Globo do bingo acho que você consegue fazer o gerador de cartelas. Basta criarmos um preenhedor de vetor.

```
void preencher_vetor(_vetor, _min, _max):
    Para cada _num entre _min e _max:
        insere _num em _vetor
```

Nossa cartela pode ser uma matriz de 5x5 de string. Agora você pergunta, porque string? Não é melhor fazer com inteiros? Lembra daquele ## no centro da cartela? Se nossa cartela for de string fica fácil colocar ele alí.

A primeira coluna da matriz corresponde ao B. A segunda corresponde ao I e etc.

Então fazemos uma funçãozinha onde dizemos a coluna e o valor min e max dela. Na função montamos um vetor com os valores válidos para aquela coluna. Então vamos para cada linha, puxando um valor do vetor e inserindo na linha.

```
void preencher_coluna(_cartela,_coluna, _min, _max):
    criar _vetor vazio
    preencher_vetor(_vetor, _min, _max)
    Para _linha de 0 a 4:
        bola = pegar_bola(_vetor)
        coloque bola em _cartela[_coluna][_linha]
```

Agora ficou ultra moleza. Montar a cartela seria só chamar a função passando os valores corretos para cada coluna.

```
void preencher_cartela(_cartela):
    preencher_coluna(_cartela, 0, 1, 15)
    preencher_coluna(_cartela, 1, 16, 30)
    ...
    //e agora fazemos
    _cartela[2][2] recebe "##"
```

Aposto que você consegue criar um laço pra isso ao invés de ir fazendo de um por um.

4 Diversão

Agora é a parte da diversão. Gere uma cartela. Depois vá chamando bolas do globo até que a cartela "bingue". E me diga quantas bolas você precisou chamar. Pra isso crie uma função bingou que retorna verdadeiro se todas as bolas de uma cartela estão na mesa.

```
    crie uma _cartela
    crie uma _mesa e um _globo:
    Enquanto !bingou(_cartela, _mesa):
        puxar bola do _globo e por na _mesa
```

```
    imprimir quantidade de bolas puxadas
```