Universidade Federal do Ceará
Campus Fortaleza - Centro de Ciências
Departamento de Computação
Curso de Ciência da Computação

Lista de Exercícios 5

Algoritmo 1 – (Extraído da Web) Bingo de Programação)

Crie um programa que implemente o jogo "Bingo de Programação". Nesse jogo, o jogador deve selecionar a quantidade de números que ele gostaria de apostar (entre 1 e 20), e em seguida, informar os números escolhidos (valores entre 0 e 100). Após receber a aposta, o computador sorteia 20 números (entre 0 e 100) e compara os números sorteados com os números apostados, informando ao apostador a quantidade de acertos e os números que ele acertou.

O seu programa deverá implementar as funções:

`ler_aposta`, `sorteia_valores` e `compara_aposta`.

A função `ler_aposta` deve receber como parâmetro a quantidade de números que serão apostados e um vetor previamente alocado dinamicamente para armazenar a quantidade exata de números apostados. A função deve pedir para o usuário digitar os números apostados e armazená-los no vetor, garantindo que somente números dentro do intervalo de 0 a 100 sejam digitados. A função deve seguir o seguinte protótipo:

```
void ler_aposta(int *aposta, int n);
```

A função `sorteia_valores` deve receber como parâmetro a quantidade de números que serão sorteados e um vetor previamente alocado dinamicamente para armazenar a quantidade exata de números sorteados. A função deve sortear aleatoriamente os números (entre 0 e 100) e armazená-los no vetor. A função deve seguir o seguinte protótipo:

```
void sorteia_valores(int *sorteio, int n);
```

A função `compara_aposta` deve receber como parâmetro o vetor com os números apostados (`aposta`), o vetor com os números sorteados (`sorteio`), juntamente com os seus respectivos tamanhos (`na` e `ns`) e um ponteiro para uma variável inteira (`qtdacertos`), onde deve ser armazenada a quantidade de acertos. A função deve retornar o ponteiro para um vetor alocado dinamicamente contendo os números que o apostador acertou. A função deve seguir o seguinte protótipo

```
int *compara_aposta(int *aposta, int *sorteio, int *qtdacertos,
                    int na, int ns);
```

Em seguida, crie a função principal do programa utilizado as funções criadas anteriormente para implementar o jogo "Bingo de Programação". Lembre-se que os vetores `aposta`, `sorteio` e `acertos` devem ser alocados dinamicamente e a memória alocada deve liberada quando ela não for mais ser utilizada.

Para sortear números aleatórios utilize a função `rand` da biblioteca `stdlib.h`. A função `rand` retorna um número aleatório em um determinado intervalo. Exemplo:

```
x = rand() % 10; /* x vai receber um valor entre 0 e 10 */
```

Para garantir que novos números aleatórios sejam sorteados em cada execução do programa é necessário executar a função `srand` antes de sortear os números. Exemplo:

```
srand(time(NULL));
```

Para poder utilizar essas funções é necessário incluir no programa as bibliotecas `stdlib.h` e `time.h`.

Exemplo de programa para sortear um número aleatório:

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

int main(void)
{
      int x;
      srand(time(NULL));
      x = rand() % 10; /* x vai receber um valor entre 0 e 10 */
      printf("%d", x);
      return 0;
}
```

Algoritmo 2) De acordo com o algoritmo 2 da lista de exercícios 2 (Lista de Alunos Aprovados e Reprovados), modifique a estrutura de armazenamento dos alunos para um vetor cujo tamanho seja alocado dinamicamente durante a execução do programa.

As alterações necessárias são as seguintes:

1) No início da execução do programa, deve ser perguntado ao usuário quantos alunos inicialmente o programa deve atender. De acordo com o número fornecido, deve ser criado dinamicamente um vetor de alunos para guardar estes dados;

2) Na inclusão de alunos, caso se tente inserir um novo aluno além do tamanho inicialmente alocado, deve ser oferecido ao usuário a opção de "dobrar o tamanho do vetor". Caso o usuário opte por dobrar, um novo vetor deve ser criado dinamicamente, com o dobro de elementos do vetor atualmente usado, e os alunos devem ser copiados para este novo vetor. O vetor antigo deve ter seu espaço de alocação devolvido ao SO. Isso permitirá que novos alunos sejam incluídos no sistema

3) Implementar uma função que recebe um vetor de alunos e um numero inteiro n, e imprime os n elementos desse vetor no formato:

Nome do Aluno: <Nome> - Matrícula: <Matrícula>

Notas: <N1>, <N2> e <N3> - Média: <(N1 + N2 + N3) / 3>

Faltas: <Faltas>

4) Modificar as funções que apresentam a lista de aprovados, reprovados por desempenho e reprovados por falta para que cada uma delas retorne um vetor contendo apenas os alunos em cada uma das situações e o total de alunos em cada situação. Ex: Se de um total de 10 alunos, 8 foram aprovados, então deve haver uma função com a seguinte assinatura:

```
<Estrutura_Aluno>  *vetor_aprovados(<Estrutura_Aluno>  *vetor_alunos,
                                    int totalAlunos, int *totalAprovados)
```

Funções semelhantes devem ser implementadas para o caso de alunos reprovados.

Ao receber o novo vetor, deve ser utilizada a função implementada no item 3 para apresentar os dados dos alunos selecionados.

Após esta listagem, o vetor deve devolver a memória alocada dinamicamente.