Engenharia de Computação – Campus Nova Gameleira
Disciplinas: Programação de Computadores I (PC I) e Lab. PC I– Turmas Extras - Semestre: 2º/2022
Prof.: Evandrino Barros (evandrino@cefetmg.br)

Lista 3 – Funções, Vetores e *Strings* - em duplas - 5 pontos – entrega somente pelo AVA.

Gerar um arquivo PDF com os códigos fontes em C dos programas solicitados abaixo, incluindo as telas de execução de cada um. Utilize os recursos do seu computador para imprimir o resultado das telas. Todos os recursos da Linguagem C para solução desse exercício são apresentados nas transparências das Aula 1 a 9 de PC I, presentes no AVA da disciplina.

A entrega deve ser feita pelo AVA da disciplina no link disponível para isso. Envie somente um PDF por dupla. Inclua o nome de cada integrante no documento. Inclusões posteriores não serão aceitas.

1) Implemente as seguintes funções que manipulam apenas valores inteiros:

   iszero – Verifica se o parâmetro *n* é ou não zero.
   add1 – Adiciona 1 ao próprio parâmetro *n*.
   zero – Coloca zero no parâmetro *n*.

2) Implemente a função **Ler_Inteiro** que devolve o inteiro lido. O parâmetro *prompt* contém a mensagem que solicita o inteiro ao usuário. Deverá receber apenas inteiros superiores ou iguais a zero.

   int Ler_Inteiro(char *prompt)

3) Transforme a função anterior de tal forma que o inteiro seja colocado num parâmetro enviado ao **procedimento** Ler_Inteiro.

4) Escreva, de forma iterativa e recursiva, a função de *Fibonacci*[5] que é definida da seguinte forma:

```
Fib(0) = 1
Fib(1) = 1
Fib(n) = Fib(n-1) + Fib(n-2)   , para n>=2
```

   Em seguida, escreva um programa que apresente o resultado da execução das duas funções na tela e verifique, comparativamente, os tempos de cálculo de ambas as funções para Fib(35) e Fib(40).

5) Implemente, de forma recursiva, a função **strcat** que concatena duas strings.

6) Implemente o utilitário **Line .c** que mostre o conteúdo de um arquivo recebido na linha de comando, mostrando para cada linha o número respectivo.

```
$ type dados.dat
As armas
Os Barões
Assinalados
Luis de Camões

$ line dados.dat
  1: As armas
  2: Os Barões
  3: Assinalados
  4: Luis de Camões
```

7) Nesta questão você implementar um programa para guardar notas de alunos de PC I em um arquivo, além de poder alterar notas, incluir novas notas e exibi-las na tela.

A seguir, apresentamos a estrutura de dados *tipo_nota*, a qual será usada no programa.

```
typedef struct struct_nota{ // nome da estrutura: struct_nota
    char aluno[20];  //nome do aluno
    int aval1;       //nota da primeira avaliação
    int aval2;       //nota da segunda avaliação
    int aval3;       //nota da terceira avaliação
    int exerc;       //nota dos exercícios
} tipo_nota;  //nome do tipo: tipo_nota
```

A seguir, listamos as funcionalidades que o programa terá.

a) O programa deve manipular até 40 notas de alunos. Ou seja, deve declarar um vetor *tipo_nota notas_pc1[40],* por exemplo. Não necessariamente deve constar 40 registros no vetor, sendo, portanto, possível trabalhar com um número menor de notas. Cada conjunto de dados de notas de um aluno (aluno, aval1, aval2, aval3, exerc) é chamado de registro e deve ser armazenado no vetor. O programa deve, permitir a inclusão, exibição de notas e alteração dos dados de um registro. O controle de quantas posições do vetor têm registros válidos é um aspecto importante na contrução e funcionamento do programa.

b) O programa deve ter um menu principal de opções para realizar as seguintes operações. As opções do menu devem ser escolhidas a partir de uma repetição até que seja pressionado 0 (zero) para sair. As opções são:
   i. Incluir registros no vetor a partir da leitura de dados pelo teclado.
   ii. Exibir as notas de um registro, o qual deve ser retornado por uma função de busca, cujos parâmetros são o vetor e o nome a ser pesquisado. Nesse caso, o retorno da função deve ser o endereço de memória onde o registro a ser alterado se encontra.
   iii. Alterar as notas de um registro, o qual deve ser alterado por uma função que, receba como parâmetros as novas notas e o nome, lidos pelo teclado. Essa função deve usar a mesma função de busca do item b) acima para obter o registro a ser alterado.
   iv. Gravar arquivo. Se o arquivo já existir, deverá se sobreposto nessa opção. Ou seja, não é para alterar o arquivo já existente, caso tenham sido alterados os registros lidos do arquivo e carregados no vetor. Se o arquivo não existir, deve ser criado e gravado com o conteúdo atual do vetor de registros.
   v. Ler o arquivo existente. Caso o arquivo exista, o programa deve carregar os dados do arquivo no vetor de registros do tipo tipo_nota.
   vi. Apresentar todas as notas de cada aluno na tela, bem como a soma das notas. A soma deve ser implementada por uma função que receba o registro e devolva as notas somadas.
   vii. Apresentar os alunos que obtiveram mais de 60 pontos e a média de toda a turma. A soma das notas deve se obtida com a função do item f) acima.
   viii. Opção 0 para sair da repetição que apresenta continuamente o menu de opção e sair do programa.

Como últimas observações:

a) O arquivo manipulado deve ser binário.
b) O programa será testado na correção com um arquivo binário gerado pelo professor/monitor com registros do tipo REGISTRO_NOTAS. Então, certifique-se que o seu programa conseguirá usar um arquivo binário de outra pessoa, evitando muita penalização na correção.