Segunda lista de exercícios v1.00
Engenharia Elétrica - Eletrônica – 2º Semestre de 2015
Prof: Daniel Rodrigo Ferraz Bonetti
SSC0300 – Linguagem de Programação e Aplicações

1. Faça um programa que leia uma string digitada pelo usuário e salve em um arquivo em branco.

2. Faça um programa que leia uma nova string digitada pelo usuário e, ao invés de salvar em um novo arquivo, salve no final do arquivo criado no Exercício 1.

3. Crie uma estrutura de dados com os seguintes campos: e-mail, senha, primeiro nome, último nome, idade e data de nascimento. Faça um programa que tenha duas funções: uma para ler os dados da struct e outra para mostrar na tela.

4. Modifique o programa do Exercício 3 para poder ler e salvar os dados da struct em um arquivo.

5. Faça um programa que pergunte ao usuário quantos registros o usuário deseja cadastrar. Em seguida, aloque a estrutura de dados do Exercício 3 dinamicamente para receber quantos registros foram fornecidos.

6. Faça um programa que salve os registros do Exercício 5 em um arquivo. A primeira linha desse arquivo deverá conter a quantidade de registros que o arquivo estará armazenando.

7. Crie uma função que leia os registros armazenados feitos no Exercício 6, salve todos os registros na memória e imprima-os na tela.

   Dica: Utilize o símbolo ‘<’ para preencher os campos da sua struct automaticamente. Por exemplo, crie um arquivo chamado entrada.txt com o conteúdo:

   rato@tatu.com.br
   ratotatu123
   Rato
   Tatu
   25
   13/11/1990

   E adicione como parâmetro na execução do seu programa assim:

   Meuprograma.exe < entrada.txt

8. Faça um programa que leia e salve o conteúdo de apenas um registro da struct definida no Exercício 3 em arquivo, no modo binário.

9. Faça um programa que leia um arquivo digitado pelo usuário e determine o número de linhas (nl) e de caracteres (nc) desse arquivo. Utilizando recursão, faça uma função que imprima na tela os valores inteiros de nl até nc.

10. Crie um programa que construa a estrutura de dados de uma fila que armazene valores reais. Implemente as seguintes operações:

    a. Adicionar item
    b. Remover item
    c. Mostrar o número de itens
    d. Mostrar o maior e menor elemento
    e. Procurar por um item
    f. Imprimir todos os itens

g. Sair

11. Substitua o tipo de dados real da fila do Exercício 10 pela estrutura do Exercício 3.

12. Faça um programa que crie uma estrutura de dados do tipo pilha que armazene apenas um caractere. Implemente as seguintes operações para sua pilha:

    a. Empilhar
    b. Desempilhar
    c. Mostrar número de itens
    d. Mostrar o maior e menor item
    e. Encontrar um item
    f. Imprimir todos os itens
    g. Sair

13. Faça um programa que leia uma fórmula digitada pelo usuário. Essa fórmula pode conter números, operadores e abre/fecha parêntesis. Utilizando uma pilha, verifique se os parêntesis estão corretos. Por exemplo, a fórmula x+(2*y) é válida, enquanto que a fórmula ((x-1) – 10 não é válida.

14. Escreva, com suas próprias palavras, como você implementaria uma fila utilizando duas pilhas?

15. Implemente uma função recursiva que computa a soma de todos os elementos abaixo de sum certo valor n passado por parâmetro.

16. Faça uma função recursiva que retorna o menor elemento de um vetor passado por parâmetro.

17. Faça um função que compute a soma de todos os elementos de um vetor, em que o vetor e seu tamanho são passados por parâmetro.

18. Faça um programa com duas funções recursivas: uma para calcular o fatorial e outra para Fibonacci.

19. Faça um programa que leia uma string digitada pelo usuário e imprima seu reverso utilizando recursão.

20. Faça um programa que calcule a potência de a^b digitados pelo usuário, utilizando recursão.

21. Faça um programa que conte quantas vezes um certo caractere c aparece em uma string s, utilizando recursão.

22. Crie uma lista encadeada simples que seja capaz de armazenar valores inteiros. Em seguida, implemente as seguintes operações para sua lista, utilizando funções:

    a) Inicializar lista
    b) Inserir ordenado na lista
    c) Remover item da lista
    d) Verificar se existe item na lista
    e) Mostrar quantidade de itens
    f) Imprimir todos os elementos da lista

23. Modifique a estrutura de dados dos nós do Exercício 22 para uma lista duplamente encadeada.

24. Crie uma nova versão do Exercício 23 alterando o dado a ser armazenado, isto é, ao invés da lista armazenar um valor do tipo inteiro, crie uma estrutura de dados com os campos utilizados no Exercício 3.

25. Faça um programa de autenticação utilizando e-mail e senha. Primeiramente, leia de um arquivo os registros dos usuários e armazene-os em memória utilizando uma lista encadeada (Exercício 7). Em seguida, peça que o usuário digite um nome e senha. Percorra a lista encadeada e verifique se existe algum registro com o nome e senha fornecidos. Se houver, imprima todos os dados do registro na tela e, caso contrário, informe que a combinação de usuário e senha estão incorretos.

26. Faça um programa que implemente as seguintes operações para um árvore binária capaz de armazenar valores inteiros:

    a. Inicializar árvore
    b. Inserir ordenado
    c. Verificar se existe item na árvore
    d. Mostrar quantidade de itens
    e. Encontrar o maior item
    f. Encontrar o menor item
    g. Mostrar a profundidade da árvore
    h. Imprimir todos os elementos nas ordens:
        i. Pré-ordem
        ii. Infixida
        iii. Pós-ordem

27. Utilizando uma árvore binária (Exercício 26), mostre o caminho que é necessário percorrer desde a raiz até um valor passado por parâmetro, mostrando os valores dos nós.

28. Se você precisasse desenvolver um programa que listasse todos os arquivos e pastas do seu computador, quais conceitos aprendidos em aula você acharia interessante utilizar? Por que?

29. Qual estrutura de dados você acharia mais apropriada para armazenar as chamadas de uma função recursiva? Por que?

30. Aloque um vetor de inteiros dinamicamente com tamanho definido pelo usuário e atribua valores aleatórios (sem repetição).

31. Implemente os métodos de ordenação abaixo e aplique no vetor alocado no Exercício 30. Verifique se todos os métodos de ordenação produziram os mesmos resultados.
    a. Bubble Sort
    b. Selection Sort
    c. Insertion Sort

32. Ordene o vetor do Exercício 30 utilizando o Quick Sort da Linguagem C. Utilize a função qsort da biblioteca stdlib.h. A definição da função qsort é: void qsort(void *base, size_t nitems, size_t size, int (*compar)(const void *, const void*)), em que base é o ponteiro do primeiro elemento do seu vetor, nitems é o tamanho do seu vetor, size é o tamanho de cada elemento (por exemplo, para um vetor de inteiros size=sizeof(int)) e compar é uma função de comparação (implementada por você) que deverá retornar um inteiro. Acesse http://www.cplusplus.com/reference/cstdlib/qsort/ para mais informações.

33. Crie uma estrutura de dados que tenha os campos ID e nome. Em seguida, crie duas funções para ordenar esse vetor. A primeira ordena pelo ID e a segunda pelo nome. Os métodos de ordenação disponíveis serão: Bubble Sort, Selection Sort, Insertion Sort e Quick Sort.

34. Qual algoritmo de ordenação você considera mais simples implementar? E qual deles você acha que é mais eficiente? Justifique sua resposta.

35. Faça um programa que determina o tempo de execução de todos os métodos de ordenação do Exercício 34. Utilize a seguinte função para medir o tempo:

```
#include “time.h”
double tempoAgoraMs() {
      clock_t start = clock();
      double s = start / CLOCKS_PER_SEC;
      return s/1000.0;
}
```

36. Considere o seguinte vetor:

20 17 49 27 32 82 76 43

Mostre como você ordenaria esse vetor passo a passo utilizando o Merge Sort. Escolha outro método de ordenação de sua preferência e mostre também como ordenaria esse vetor passo a passo.

37. Mostre passo a passo como ficaria uma árvore binária ordenada após a inserção dos elementos abaixo e responda: Qual a raiz, altura, profundidade, as folhas, e grau da árvore resultante? A árvore resultante é balanceada e completa?

40 45 12 95 2 48 10 11 17 42 46

Aviso: (opcional) Você pode criar apenas um projeto em sua IDE preferida (Visual Studio, Codeblocks, Dev etc) e, dentro da função main(), criar uma função para cada exercício. Ex:

```
void exer01() {
    // conteúdo do Exercício 01 aqui
}

void exer02() {
    // conteúdo do Exercício 02 aqui
}

void exer03() {
    // conteúdo do Exercício 03 aqui
}

.
.
.

void exerN() {
    // conteúdo do Exercício N aqui
}

int main() {
    // exer01(); // Você pode comentar os exercícios que já rodou
    // exer02();
    // exer03();
    ...();
    exerN();
```

```
}
```

Boa sorte ;)