Universidade Federal de Santa Maria
Prof. Cesar Tadeu Pozzer
Disciplina: Lógica e Algoritmo - ELC1064
20/02/2013

Exercícios de Linguagem C

Implemente soluções para os seguintes problemas. Teste as soluções em algum compilador C.

1. Estruturas

1) Crie estruturas, com possíveis hierarquias, para conter:
   - Nome de uma pessoa, endereço (número, rua, cidade, estado, país), banco (número da conta, quantia em dinheiro), nomes dos 5 filhos.
   - a definição de um conjunto de 3 coordenadas que definem um ponto no espaço 3D, que também possui uma cor associada, no formato R,G,B.
   - Definição de uma esfera que possui centro e raio.
   - Cubo definido por planos definidos por 4 pontos 3D.
2) Crie estruturas hierárquicas para definição de uma bicicleta. Ela deve ter duas rodas, rosetas do pedal e da roda traseira (roseta é uma roda dentada aonde vai a correia), bloco e direção. Tendo esta estrutura definida, faça uma função que determina qual a distância entre as duas rodas e outra função que determina qual das 4 "rodas" tem maior raio.
3) Mostre vantagens da organização dos dados de forma hierárquica usado estruturas aninhadas.
4) De forma semelhante à definição de uma hierarquia de estruturas para definição de uma bicicleta, descreva em C, estruturas para dar suporte aos seguintes problemas:
   - motor (parafuso, marca, dimensões, eixo, material de cada parte, ...)
   - carro (motor, potência, velocidade, ...)
   - casa (tipo de madeira usada para porta e janela, definição dos quartos, sala, proprietário, etc...)
   - cadastro de pessoas (idade, sexo, altura, profissão,...)
   - cadastro de produtos de um mercado (data de validade, preço, nome, ...)
   - primitivas geométricas (ponto, linha, circulo, ...)
   - relógio (engrenagens, hora atual, ponteiros, ...)

2. Ponteiros

1) Explique o que é:
   - um ponteiro
   - um ponteiro para ponteiro
   - o endereço de um ponteiro
   - o conteúdo de um ponteiro
   - o endereço de uma variável
2) Explique o que acontece quando um ponteiro aponta para uma variável. Usar uma representação gráfica da memória.
3) Assuma a seguinte definição:

```
int a, b;
int *P1, *P2, **P3;
float *P4;
```

Diga quais das sentenças são verdadeiras e quais são falsas (justifique)

• a = 10	• P4 = P1
• b = &a	• P2 = P1 = &a
• P1 = a	• *P1 = 20
• a = &P1	• *P2 = *P1
• *P1 = &a	• *P3 = &P1

• &P1 = &a	• P3 = &P2
• P4 = &a	• **P3 = *P1

4) Explique a diferença entre passagem de parâmetros por valor e por referência. Implemente algumas funções de potenciação que usam as duas estratégias.
5) O que pode acontecer se for atribuído algum valor a um ponteiro que não tenha sido inicializado. Ex:
```
float *p;
*p = 2000;
```
6) Em C, não se pode fazer uma função que retorne dois valores, exceto com o uso de estruturas. Uma forma de solucionar esta restrição é com o uso de ponteiros, pois a função pode receber qualquer numero de variáveis por referência. Faca programa, que possuindo duas variáveis inteiras a=2 e b=3, chame a função void troca(int *a1, int *b1), que deve fazer a inversão dos valores de a e b, ou seja, b passa a valer o que a valia e a passa a valer o que b valia. Apos a chamada da função troca(), imprimir os valores de a e b. A função troca também deve ser implementada, e deve ter tipo de retorno void.
7) Declare vetores de inteiro, char, float, double, long int com 5 posições, da seguinte forma:
```
int   v[5] = {2,5,1,4,0};
char  c[5] = {'a','b','m', '4','-'};
float v[5] = {2.66, 0.125, 1.0, 4.99, 2.009};
```
Usando a função printf, com o argumento "%p", e com vetores apontando para cada um dos tipos de dados, descubra quantos bytes é alocado pelo seu compilador a cada tipo de dados. Não se esqueça que para um vetor apontar para um vetor de float, ele deve ser do tipo float. Como se sabe que em um vetor as posições são contínuas, se for impresso o endereço de duas posições, pela diferença entre os dois endereços pode-se descobrir quantos bytes são alocados.
8) Declare um vetor de inteiros com 300 posições, de forma que cada posição possua o valor igual o índice da posição (logo, o vetor será ordenado de 0 a 299). Declare um ponteiro que aponte para a quinta posição deste vetor, ou seja,
```
p = &v[5];
```
Usado a função printf, e o endereçamento do tipo *(p ± n), imprima o valor de todos os elementos do vetor, com respectivos endereços. Não confundir com o endereçamento *p+n, que imprime o conteúdo de p somado a n.
9) Explique o que faz o seguinte trecho de código
```
float f = 10.5;
int i = *(int*)&f;
```

3. Strings

1) Assumindo a seguinte declaração:
```
char v[10] = "123454321";
```
Faça uma função genérica que recebe e imprime a string, usando laços de repetição e endereçamento de ponteiros (*(n ± n)), da seguinte forma:
```
5
454
34543
2345432
123454321
```
A função de impressão deve localizar o meio da string, e deve ser genérica a qualquer tamanho de string. Para descobrir o tamanho da string, use a função strlen( ), que está definida em string.h.
2) Implemente funções que façam o mesmo que as funções do C:
   - strlen( ) - retorna o tamanho da string
   - strcmp( ) - diz se as duas strings são iguais ou não
   - strncpy( ) - copia os n primeiros caracteres para a string destino
   - strcat( ) - concatena duas strings
   - strfind( ) - procura todas as ocorrência de uma string dentro de outra. Deve informar todas as posições iniciais onde ela encontrou.
4) Função que recebe uma string e imprime o valor ASCII de cada elemento.

5) Função que procura a ocorrência de uma substring dentro de outra string. Todas ocorrências da substring devem ser impressas na tela, incluindo a posição do vetor na string maior que corresponde a primeira letra da string encontrada. Levar em consideração letras maiúsculas e minúsculas. Ex: procurar a ocorrência de "os" dentro da string "os ponteiros sao uteis". Neste caso deve encontrar duas ocorrências, nas posições 0 e 10.
6) Fazer a inversão de uma string (ordem de elementos). Ex: "abc" → "cba"
7) Supondo que uma string contenha o nome de um arquivo qualquer de imagem no formato gif. Faça uma função para alterar a substring "gif" por "bmp". Neste caso deve-se localizar a posição do ponto e substituir desta posição em diante.
   Ex: foto1.gif → foto1.bmp
8) Função que gere uma sequência de M nomes de arquivos que iniciem com uma palavra específica e terminem com uma numeração de N dígitos, seguido de uma extensão. A função deve receber o nome base, o número de dígitos e o tipo da extensão. Ex: `gera_arquivo("dado", 3, 5, "dat")` deve imprimir na tela o seguinte resultado: dado001.dat, dado002.dat, dado003.dat, dado004.dat, dado005.dat

4. Arquivos Texto (funções fopen, fclose, fscanf, fprintf, fgetc, fgets, fputc)

1) Faça um programa para ler um arquivo texto (o usuário deve fornecer o nome do arquivo) e imprimir seu conteúdo na tela. Para controlar a velocidade de impressão, o programa deve imprimir uma linha ou uma tela por vez.
2) Programa que lê um arquivo texto e copie apenas os caracteres alfabéticos (letras) para um arquivo de destino. Números e caracteres especiais devem ser desconsiderados.
3) Programa que procura pelas ocorrências de uma *string* dentro de um arquivo texto e informe em que posições (linha, coluna) formam encontradas as ocorrências.
4) Faça uma função que gere um arquivo texto com N linhas e M colunas, onde cada valor numérico é um valor inteiro randômico. A separação entre uma coluna e outra deve ser feita por um ou mais espaços em branco. Faça outra função para ler e imprimir o arquivo gerado.
5) Faça um programa para ler um arquivo texto que possui a seguinte estrutura: um identificador indicando o número de linhas de dados do arquivo, seguido dos dados, organizados em 3 colunas, sendo a primeira um caractere, seguindo de um valor inteiro e um valor real. Ex:

```
4
a  30  12.6
v 4      8.88804
  b   5555  0.0001
x 123456  1
```

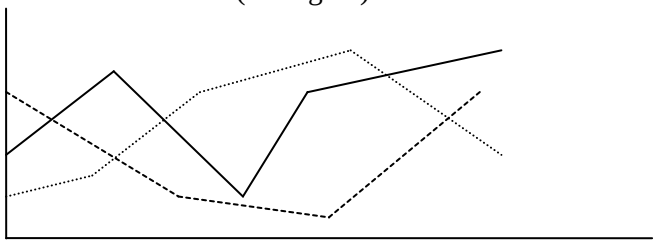
6) Tendo-se um arquivo texto, que possui seus dados (numéricos) dispostos em 4 colunas de valores inteiros, faça um programa que imprima na tela apenas o valor de uma coluna especificada pelo usuário. O programa poderá também gravar esta coluna em outro arquivo de saída.
7) Programa para ler um arquivo de dados numéricos do tipo float, dispostos em uma coluna, e dizer em que linha foi encontrado o maior valor.
8) Dado um arquivo numérico com duas colunas de valores inteiros, fazer uma função que leia estes dados e gere um arquivo com 3 colunas, sendo a terceira coluna o valor da soma das outras duas. O número de linhas do arquivo deve permanecer o mesmo.
9) Escreva um programa para fazer um parser em arquivos do tipo XML.
10)

5. Arquivos Binários (funções fwrite, fread, fseek)

1) Faça um programa que simule um controle de estoque de uma loja, onde cada produto, representado por meio de um registro, possui um identificador inteiro, nome, quantidade e custo unitário. O programa deve permitir a inclusão e remoção de novos produtos, consulta de produtos por nome, alteração de

registros, geração de relatórios (ex: qual vendeu mais, qual tem maior estoque, produtos cujo estoque estejam abaixo de X unidades, etc.). Todos os registros devem ser armazenados seqüencialmente em um arquivo binário. O programa deve disponibilizar um menu de opções ao usuário. Ex: 1 – Incluir Produto, 2 – consultar, etc.

2) Programa que lê um arquivo com 5 colunas de dados numéricos e um número qualquer de linhas e gera um gráfico na tela referente aos 5 valores de cada linha. Cada gráfico deve ter uma cor diferente. Os gráficos podem se cruzar. Este tipo de gráfico pode ser muito útil para fazer comparações entre valores das colunas. Ex: Variação da intenção de voto nos meses de campanha eleitoral referentes aos candidatos da cidade. (ver figura)



aqui pode ter uma legenda dos meses

3) Implemente uma função que lê um arquivo em formato BMP, faça a inversão da imagem no eixo x (flip) e grave a imagem modificada em um arquivo de destino.