MAC122 – Princípios de Desenvolvimento de Algoritmos

DCC - IME - USP
2º Semestre de 2014
Prof.: Dr. Paulo Miranda & Marco Aurélio Gerosa
pmiranda@vision.ime.usp.br, gerosa@ime.usp.br

Lista 01:
Funções & Strings

Implemente as funções especificadas abaixo:

OBS: Não é para usar as funções do <string.h>. Você só pode usar as funções que tiver implementado. Por questões de simplicidade, assuma o padrão ASCII.

1) `void InverteString(char str[]);`
Altera a própria string **str** de modo que ela fica invertida.
Exemplo: Se inicialmente temos:
str=`"PAMELA SILVA"`, no final teremos:
`"AVLIS ALEMAP"` em **str**.

2) `void InvertePalavras(char str[]);`
Altera a própria string **str** de modo que todas as suas palavras individualmente ficam invertidas.
Exemplo: Se inicialmente temos:
str=`"PAMELA SILVA"`, no final teremos:
`"ALEMAP AVLIS"` em **str**.

3) `void ReduzEspacos(char dest[], char src[]);`
Gera em **dest** uma cópia de **src** porém eliminando o excesso de espaços em branco, de modo que todas as sequências de espaços em branco consecutivos são convertidas em um único caracter de espaço. Assuma que o vetor **dest** é grande o suficiente para conter o resultado.
Exemplo: Se inicialmente temos:
src = `"  Prova  P1  MAC122  "`, no final teremos:
dest = `" Prova P1 MAC122 "`.

4) `void ProcuraEmail(char dest[],char src[]);`
Busca em **src** por possíveis candidatos a endereços de e-mail. Copia em **dest** a primeira sequência de caracteres não brancos encontrada que contenha o caracter '@', ou uma string vazia caso contrário.
Assuma que o vetor **dest** é grande o suficiente para conter a palavra encontrada.
Exemplo:
Para **src**=`"Enviar mensagem para lucas@ig.com.br ou para outro moderador"` temos `"lucas@ig.com.br"` que ficará em **dest**.

Alocação Dinâmica & Strings

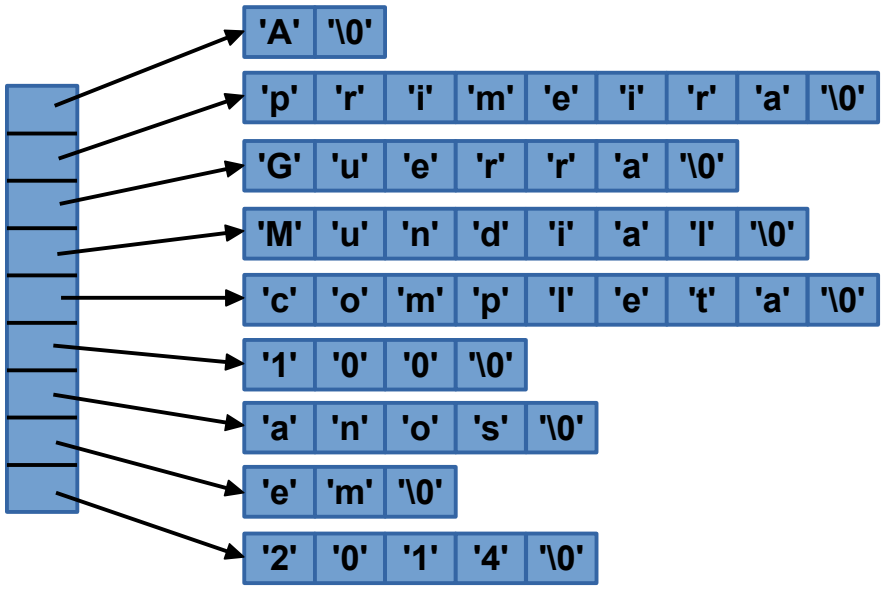
Para uma dada string, contendo uma linha de texto, faça uma função em C que separa cada palavra do texto:
A função deve devolver o endereço de um vetor de apontadores criado dinamicamente com tamanho igual ao número de palavras do texto. Os apontadores desse vetor devem apontar para cópias das palavras do texto. Cada cópia deve ser amazenada em um vetor alocado dinamicamente, usando a menor quantidade de memória possível.

Use o protótipo abaixo:

```
char **VetorDePalavras(char texto[], int *npal);
```

onde npal aponta para uma variável que irá guardar o número de palavras encontradas.

Exemplo:
Para o texto de entrada: `"A  primeira Guerra Mundial  completa 100 anos  em 2014"`, no final teremos a seguinte estrutura alocada:



Alocação Dinâmica de Memória

1) Faça uma função em C que construa dinamicamente um vetor **B** contendo os índices dos elementos ímpares de um primeiro vetor **A** fornecido como parâmetro. Exemplo:
   Para **`A = (1 4 6 7 9 8)`**, a função deve devolver **`B = (0 3 4).`**

   Use o protótipo abaixo:

   ```
   int *IndicesImpares(int A[], int nA, int *nB);
   ```
   onde nB aponta para uma variável que irá guardar o tamanho do vetor B alocado.

2) Dada a seguinte definição de estrutura:

   ```
   struct Vetor{
        int n;
        int *valor;
   };
   ```

   Faça uma função que aloque dinamicamente uma estrutura do tipo **`struct Vetor`**, sendo que o seu campo **`valor`** deve apontar por sua vez para um vetor de inteiros de comprimento **`n`** também alocado de modo dinâmico.
   Use o protótipo abaixo:

   ```
   struct Vetor *AlocaVetor(int n);
   ```

3) Refaça a função do item 1, porém agora usando o seguinte protótipo:

   ```
   struct Vetor *IndicesImpares(struct Vetor *A);
   ```

   Note que nesse caso o tamanho do vetor **B** devolvido já está dentro da estrutura.

Alocação Dinâmica & Matrizes

1. Faça uma função em C que recebe uma matriz de inteiros **A**, com **m** linhas e **n** colunas, contendo valores no intevalo de 0 a 255, e que devolve o seu histograma, com as frequências (número de ocorrências) para cada um dos valores do intervalo. O histograma deve ser representado como um vetor de inteiros de 256 posições alocado dinamicamente. Use o protótipo abaixo:
   `int *Histograma(int **A, int m, int n);`

   Exemplo: Para a matriz $\begin{bmatrix} 1 & 2 & 3 & 5 \\ 3 & 5 & 2 & 255 \\ 255 & 2 & 5 & 2 \end{bmatrix}$

   O histograma deve ser: **`[0, 1, 4, 2, 0, 3, 0, …, 0, 2]`**

2. Faça uma função em C que recebe uma matriz de inteiros **A**, com **m** linhas e **n** colunas, alocada dinamicamente (conforme a representação mostrada em aula, onde cada linha é armazenada em um vetor separado), e que altera a própria matriz **A** de modo a inverter suas linhas na vertical (Flip Vertical). Use o protótipo abaixo:
   `void FlipVertical(int **A, int m, int n);`

   Exemplo: antes $\begin{bmatrix} 5 & 3 & 7 \\ 3 & 8 & 2 \\ 12 & 6 & 0 \\ 4 & 10 & 9 \end{bmatrix}$ depois $\begin{bmatrix} 4 & 10 & 9 \\ 12 & 6 & 0 \\ 3 & 8 & 2 \\ 5 & 3 & 7 \end{bmatrix}$

3. Usando a mesma representação de matriz do item anterior, agora faça a inversão dos elementos na horizontal (espelhar horizontalmente). Use o protótipo abaixo:
   `void FlipHorizontal(int **A, int m, int n);`

4. Uma matriz é triangular inferior se ela é quadrada e todos os seus elementos acima da diagonal principal são nulos. Exemplo: $\begin{bmatrix} 5 & 0 & 0 & 0 & 0 \\ 3 & 8 & 0 & 0 & 0 \\ 0 & 6 & 4 & 0 & 0 \\ 2 & 10 & 9 & 1 & 0 \\ 8 & 7 & 1 & 3 & 8 \end{bmatrix}$

   Na representação usual existe um desperdício de memória de $(n-1)n/2$ elementos nulos, onde n é a ordem da matriz. Uma representação linearizada alternativa em um vetor com $n(n+1)/2+1$ elementos é mostrada abaixo, onde os valores nulos acima da diagonal são representados uma única vez na primeira posição do vetor:
   `[0, 5, 3, 8, 0, 6, 4, 2, 10, 9, 1, 8, 7, 1, 3, 8]`
   Escreva uma função que converte os índices (i,j) de um elemento da matriz usual para o seu índice correspondente na representação linearizada.

Estruturas & arquivos (modo texto)

1.a:
Um triângulo pode ser representado pelas coordenadas reais (float) dos seus três vértices **(Xa, Ya)**, **(Xb, Yb)**, e **(Xc, Yc)**. Escreva uma possível definição em C para as estruturas de um vértice e de um triângulo, e implemente as seguintes funções em C:

`float Distancia(struct Vertice A, struct Vertice B);`
Calcula a distância euclidiana entre os vértices (pontos) A e B.
OBS: Use a função `float sqrtf(float x);` para calcular a raiz quadrada.

`float Perimetro(struct Triangulo T);`
Calcula o perímetro do triângulo T.

1.b:
Dado um arquivo texto "entrada.txt" que possui em sua primeira linha um inteiro que indica a quantidade de triângulos, e nas demais linhas os dados dos triângulos (um triângulo por linha segundo o formato: "**Xa Ya Xb Yb Xc Yc"**), conforme o exemplo:

```
5
0.0  0.0  0.0  1.0  1.0   0.0
0.0  0.0  0.0  0.1  0.1   0.0
2.0  2.0  3.0  2.0  2.5   3.0
2.0  2.0  2.1  2.0  2.05  2.1
8.0  1.0  7.0  3.0  2.0   1.0
```

Escreva um programa que:

- **Aloque dinamicamente um vetor de estruturas (triângulos) e preencha-o com os valores lidos do arquivo "entrada.txt".**

- **Crie um arquivo texto "saida.txt" que contém somente os dados dos triângulos com perímetro > 0.5.**
  Obs: A primeira linha do arquivo deve conter o número de registros e as demais os dados (um registro por linha).

- **Desaloque o vetor e feche os arquivos.**

Exemplo do arquivo "saida.txt":

```
3
0.00 0.00 0.00 1.00 1.00 0.00
2.00 2.00 3.00 2.00 2.50 3.00
8.00 1.00 7.00 3.00 2.00 1.00
```

Estruturas ligadas

Dadas as definições abaixo:

```
struct node {
  int item;
  struct node *prox;
};

typedef struct node* lista;
```

Explique o funcionamento da função XXXX abaixo, dado que ela recebe o endereço do primeiro nó de uma lista encadeada simples cujo último nó aponta para NULL:
OBS: Faça desenhos ilustrando exemplos do progresso do algoritmo a cada iteração do laço.

```
lista XXXX(lista x) {
    lista t, y, r;
    y = x;
    r = NULL;
    while (y != NULL) {
       t = y->prox;
       y->prox = r;
       r = y;
       y = t;
    }
    return r;
}
```

Exemplo de chamada dessa função:

```
int main(){
   lista p = NULL;
   lista t;
   int i;

   /* adiciona alguns elementos na lista p */
   for(i = 3; i >= 0; i--){
      t = (lista)malloc(sizeof(struct node));
      t->item = i;
      t->prox = p;
      p = t;
   }

   p = XXXX(p);

   ImprimeLista(p);

   return 0;
}
```