Departamento de Ciência da Computação – IME-USP

MAC 115 – Introdução à Computação para Ciências Exatas e Tecnologia

Escola Politécnica – Primeiro Semestre de 1997

Quarto Exercício-Programa — Data de entrega: até 19 de junho de 1997.
Turmas: Civil B, Mecânica, Naval e Química

Um formatador de textos simplificadíssimo

Um **formatador de textos** é um programa cuja entrada consiste de um texto formado por: palavras, espaços em branco, vírgulas, pontos, ..., e comandos para o formatador. As palavras são separadas por um ou mais espaços em branco e a saída consiste do texto de entrada editado como, por exemplo, este que você esta lendo.

O objetivo deste exercício programa é programar um formatador de textos simplificado com as características que serão descritas a seguir. **Neste EP você deverá usar as funções que construiu para o terceiro exercício-programa**.

O que o seu programa deve fazer

A principal tarefa do seu formatador é consumir um arquivo de entrada (contendo o texto a ser formatado) e construir uma saída em que o texto apareça formatado seguindo as diretrizes descritas a seguir.

Considere as seguintes definições em C:

```
#define MAX_TAM_LINHA_ENTRADA 100
#define MAX_TAM_LINHA_SAIDA    65
#define MAX_NUM_LINHAS_SAIDA  500

char linha_entrada[MAX_TAM_LINHA_ENTRADA];
char texto_saida[MAX_NUM_LINHAS_SAIDA][MAX_TAM_LINHA_SAIDA];
```

A constante MAX_TAM_LINHA_ENTRADA define o tamanho máximo de uma linha do arquivo de entrada. A constante MAX_TAM_LINHA_SAIDA define o tamanho máximo de cada linha da matriz texto_saida. Esta matriz é definida como tendo MAX_NUM_LINHAS_SAIDA linhas.
O seu formatador deve ler e imprimir linha a linha o texto de entrada e armazenar a linha lida na variável linha_entrada[1]. O texto que está no vetor linha_entrada deve então ser transferido para a matriz texto_saida obedecendo as seguintes regras:

1. para cada seqüência consecutiva de brancos no texto, coloque um só branco na matriz Texto.
2. O texto de entrada pode conter dois caracteres de controle: '&' e '@'. Estes dois elementos da entrada não devem ser colocados na matriz texto_saida, pois eles são comandos de controle sobre o funcionamento do formatador:
   - O controlador '&' indica novo parágrafo. Ou seja, quando este delimitador for encontrado no texto preenche-se o restante da linha com brancos (' '), muda-se para a linha seguinte continuando a preencher a matriz Texto a partir da quinta coluna dessa nova linha com o restante do texto que esta em Vetor. Note que as posições 'puladas' da matrix Texto deverão ser 'branqueadas'.

[1] Você pode usar outros nomes para as variáveis, se desejar.

- O controlador '@' indica que o restante da linha de entrada onde o '@' apareceu deve aparecer na saída de forma "centralizada", isto é, com o mesmo número de brancos de cada lado. Além disso uma linha deve ser pulada e um parágrafo começado.
  Você pode assumir que o texto a ser centralizado cabe em uma linha da matriz de saída.

Observe o seguinte exemplo de uso dos controladores:

```
Inicio do texto. & Novo paragrafo. @Titulo bem     bobo.
& Paragrafo apos titulo.
```

após formatado fica:

```
....*....1....*....2....*....3....*....4....*....5....*....6....*
Inicio do texto.
     Novo paragrafo.
                         Titulo bem bobo.

     Paragrafo apos titulo.
```

3. Quando uma palavra não cabe em uma linha deve-se tentar usar pelo menos as duas regras abaixo para separar esta palavra:

   (a) Primeira regra: vogal | consoante vogal. Pode-se separar uma palavra numa vogal que antecede uma consoante seguida de uma vogal. Exemplos: vo-gal, pa-lavra, progra-ma, cha-to.

   (b) Segunda regra: $\text{consoante}_1 \mid \text{consoante}_2$. Se $\text{consoante}_1 = \text{consoante}_2$ então pode-se separar a palavra no ponto indicado acima. Exemplos: car-ros, pas-seio, cor-reio.

   Deve-se colocar na linha corrente o maior trecho inicial possível da palavra seguida do traço ('-') separador. Eventualmente, podem sobrar alguns brancos no fim desta linha. Adiante explicaremos o que fazer nesse caso.

   Observação: É opcional a inclusão de outras regras de separação.

4. É necessário alinhar-se o lado direito de cada linha da matriz texto_saida. Para isso espalhe os brancos que sobram no fim da linha entre os brancos da mesma. Isto só deve ser feito com linhas que não são final de parágrafo.

Ao final o seu programa deve imprimir a matriz texto_saida linha por linha.

Funções

A sua tarefa fica bem mais simples com a funcionalidade esperada sendo separada em funções. O seu programa deve utilizar (**obrigatoriamente**) as funções descritas a seguir. (Você pode fazer mais funções ou procedimentos se julgar conveniente. Você vai precisar das funções do EP anterior, como espalha_branco, centraliza, e_vogal, etc).

int separa (char linha[ ], int pos, int tam_palavra, int espacos)

Esta função recebe um vetor de caracteres *linha*, um inteiro *pos* representando o início da palavra a ser separada ($0 \leq pos <$ MAX_TAM_LINHA_ENTRADA), um inteiro *tam_palavra* indicando o tamanho da palavra a ser separada e um inteiro *espacos* indicando qual o número de posições disponíveis (ou seja, espaço sobrando no final da linha de texto_saida que está sendo montada). A função retorna o comprimento do maior trecho inicial possível (menor que *espacos*) da palavra de comprimento *tam_palavra* que se inicia em *linha*[*pos*] e que é uma separação possível da palavra. Lembre-se que das espaco posições fornecidas uma deve sere reservada para o traço ('-') separador. Exemplos:

- se a palavra é 'comprimento' e espacos = 8 então a função deve devolver 6 (compri-mento);
- se a palavra é 'carro' e espacos = 3 então a função deve devolver 0;
- se a palavra é 'carro' e espacos = 4 então a função deve devolver 3 (car-ro);

void avanca_linha (char texto[ ][NUM_MAX_COLUNAS_SAIDA],
int *linha, int *coluna, int novo_paragrafo)

A função recebe uma matriz *texto* e avança para a próxima linha. Ou seja, ela preenche o restante do vetor com brancos e identa na margem direita se necessário, ou seja, se estamos no meio de um parágrafo. Esta informação é fornecida pela variável *novo_paragrafo*, que tem 1 se o parágrafo corrente terminou e 0 caso contrário. No caso de um novo parágrafo, 5 brancos iniciais devem ser inseridos. Além disso, a rotina deve atualizar os valores de *linha* e *coluna*, que indicam a nova posição no texto sendo montado.

Leitura dos dados

Os dados do seu programa devem ser lidos de um arquivo em disco. Utilize a função abre_arq_entrada definida no Terceiro EP para abrir o arquivo. O seu programa deverá ir executando enquanto existirem linhas no arquivo de entrada. Um jeito de fazer isto é:

```
FILE *arq_entrada;
int tam;
/* aqui provavelmente virão declarações de outras variáveis
 * de seu programa e inicializações de variáveis
 */
while (fgets(linha_entrada, NUM_MAX_COLUNAS_ENTRADA, arq_entrada) != 0) {
  tam = strlen(linha);     /* tam recebe o tamanho da linha efetivamente
                            * lida */
  if (linha[tam-1] == ´\n´)
    /* o comando fgets le tambem o caractere de mudanca de linha.
     * E´ bom elimina-lo, afinal ele nao faz parte do texto a ser
     * formatado
     */
    tam--;

    /* a partir daqui voce pode escrever a parte que vai percorrendo
     * a variavel linha_entrada e passando o seu texto para a matriz
     * de saida. Observe que a variavel tam contem o tamanho da linha.
     */
    ...
}   /* fim do while (fgets ...) */
```

A rotina de biblioteca fgets lê uma *string*, retornando 0 quando não há mais linhas para leitura. A função strlen calcula o tamanho de uma *string*.

Um exemplo

Teste seu programa pelo menos no seguinte exemplo:

```
@Livro de Goodis vira filme
Tocando seu piano no fundo de um bar,  Eddie renunciara
a tudo: ao  estrelato,  ao amor e
ao caminho criminoso dos irmaos.  Mas, por menos
  que quisesse,  alguma forca fazia renascer nele a paixao
 e  a  violencia.  & Num suspense angustiado e pulsante,
envolvendo crime,  violencia e um lirismo seco,
Goodis cria um romance tenso,  sempre prestes a explodir em
momentos-limite. A rua, a
neve  , as personagens desfilam lentamente
  pelos    nossos  sentidos   ,
fazendo-nos ouvir a musica de Eddie e sentir a neve nos
  acoitando o rosto  .  &
Aproveitando toda a forca visual do livro  ,
Francois  Truffault adaptou-o para o cinema  ,
 num filme pouco visto no Brasil.
```

Uma possível saída do seu programa

```
....*....1....*....2....*....3....*....4....*....5....*....6....*
                      Livro de Goodis vira filme

    Tocando seu piano no fundo de um bar, Eddie renunciara a  tu-
do: ao estrelato, ao amor e ao caminho criminoso dos irmaos. Mas,
por menos que quisesse, alguma forca fazia renascer nele a paixao
e a violencia.
    Num  suspense  angustiado  e pulsante, envolvendo crime, vio-
lencia e um lirismo seco, Goodis cria um  romance  tenso,  sempre
prestes a explodir em momentos-limite. A rua, a neve , as  perso-
nagens desfilam lentamente pelos nossos  sentidos  ,  fazendo-nos
ouvir a musica de Eddie e sentir a neve nos acoitando o rosto .
    Aproveitando toda a forca visual do livro  ,  Francois  Truf-
fault adaptou-o para o cinema , num filme pouco visto no Brasil.
```

Observe no exemplo acima todo conjunto de símbolos, diferentes de branco, que estão juntos na entrada, permanecem juntos na saída, por exemplo:

```
  bar,            estrelato,     irmaos,
  Mas,            quisesse,      violencia.
  pulsante,       seco,          momentos-limite.
  fazendo-nos     Brasil.
```

IMPORTANTE: Todo exercício-programa deve seguir as observações contidas no folheto distribuído juntamente com o primeiro EP. Neste folheto estão descritas as diretrizes para forma de entrega do exercício, aspectos importantes na avaliação, etc. Por exemplo, não esqueça de identar o seu programa e de entregar uma cópia impressa de uma execução do seu programa.