MC202 — ESTRUTURAS DE DADOS

Laboratório 03 — Calculadora de Strings

Tarefa

Nessa tarefa o aluno deverá implementar uma calculadora de “strings” em C. Como em uma calculadora comum, a nossa calculadora mantém uma string atual, chamada de buffer, e recebe uma sequência de operações a serem realizadas sobre essa string. A cada operação, a calculadora imprime uma linha com uma saída e possivelmente atualiza o buffer. A calculadora deve implementar os seguintes comandos:

CAT str	Concatena a string informada pelo parâmetro “str” com a string no buffer de forma que a string resultante seja a que estava no buffer seguida pelo parâmetro. O buffer deve ser substituído pela string resultante e uma linha de saída deve ser impressa contendo a nova string no buffer.
CMP str	Compara a string informada pelo parâmetro “str” com a string no buffer e imprime: ● IGUAL caso elas sejam iguais; ● DIFERENTE caso elas sejam diferentes.
SUB x tam	Encontra a substring do buffer que começa na posição “x” e tem tamanho “tam”. Deve-se usar 0 como posição inicial da string. A string do buffer deve ser substituída pela substring encontrada e depois se deve imprimir a string encontrada. Você pode presumir que os parâmetros x e tam são válidos.
END	Termina a execução. Nada precisa ser impresso.

O programa deve começar com um buffer vazio que será usado em cada comando. Para fazer operações sobre uma outra string, pode-se inicializar um buffer vazio através do comando CAT.

Você pode assumir que o primeiro comando sempre será um CAT.

Após a impressão de cada saída deve ser impresso também uma quebra de linha (‘\n’).

Todas as palavras informadas na entrada terão tamanho menor do que 100 e o tamanho máximo ocupado do buffer também será 100 (não será enviada nenhuma sequência de comando que exija um buffer maior que 100).

Exemplo

No exemplo abaixo, os retângulos internos correspondem à saída que seu programa deve gerar:

```
CAT casa
casa

CMP caba
DIFERENTE

CAT coisa
casacoisa

SUB 3 4
acoi

CMP acoi
IGUAL

END
```

A seguir, passamos uma pequena descrição do que é feito em cada passo, com o estado do buffer ANTES de executar o comando recebido.

- BUFFER: "" - A primeira instrução vai preencher o buffer com a palavra "casa".
- BUFFER: "casa" - Comparamos o buffer com a string "caba", que é diferente. O buffer permanece inalterado.
- BUFFER: "casa" - Concatenamos o buffer com a string "coisa".
- BUFFER: "casacoisa" - O comando sub pede para pegar a substring começando pela posição 3 (segundo 'a' de "casa") com tamanho 4.
- BUFFER: "acoi" - O próximo comando pede para comparar o buffer com "acoi", que é igual. O buffer permanece inalterado.
- BUFFER: "acoi" - Recebe o comando que finaliza a execução.

Dicas

Segue um exemplo incompleto de como tratar a entrada. A função `string_igual` retorna 1 quando a string do primeiro parâmetro igual à string do segundo e retorna 0 caso contrário. Essa função deve ser implementada por você, pois também é usada pelo comando CMP.

```
while (1)
{
   scanf("%s", comando);
   if (string_igual(comando, "CAT"))
   {
      ...
   }
   else if (string_igual(comando, "SUB"))
   {
      int x, tam;
      scanf("%d %d", &x, &tam);
      ...
   }
   ...
   else if (string_igual(comando, "END"))
   {
      break;
   }
}
```

Critérios específicos

- Os comandos devem ser implementadas sem o uso de nenhuma biblioteca externa (a não ser `stdio.h` para imprimir os resultados).
- Para as turmas E e F, este laboratório tem peso 1.
- Para as turmas G e H, este laboratório tem peso 1.
- Deverá ser submetido o seguinte arquivo: **lab03.c**.
- Tempo máximo de execução: 2 segundos.

Testando

Para compilar com o Makefile fornecido e verificar se a solução está correta basta seguir o exemplo abaixo.

```
make
./lab03 < arq01.in > arq01.out
diff arq01.out arq01.res
```

onde `arq01.in` é a entrada (casos de testes disponíveis no SuSy) e `arq01.out` é a saída do seu programa. O Makefile também contém uma regra para testar todos os testes de uma vez; nesse caso, basta digitar:

```
make testar_tudo
```

Observações gerais

No SuSy, haverá 3 tipos de tarefas com siglas diferentes para cada laboratório de programação. Todas possuirão os mesmos casos de teste. As siglas são:

1. **SANDBOX:** Esta tarefa serve para testar o programa no SuSy antes de submeter a versão final. Nessa tarefa, tanto o prazo quanto o número de submissões são ilimitados, porém arquivos submetidos aqui **não serão corrigidos**.

2. **ENTREGA:** Esta tarefa tem limite de **uma única submissão** e serve para entregar a versão final dentro do prazo estabelecido para o laboratório. Não use essa tarefa para testar o seu programa: submeta aqui quando não for mais fazer alterações no seu programa.

3. **FORAPRAZO:** Esta tarefa tem limite de **uma única submissão** e serve para entregar a versão final após o prazo estabelecido para o laboratório, mas com nota reduzida (conforme a ementa). O envio nesta tarefa irá substituir a nota obtida na tarefa ENTREGA apenas se o aluno tiver realizado as correções sugeridas no feedback ou caso não tenha enviado anteriormente em ENTREGA.

Observações sobre SuSy:

- Versão do GCC: C-ANSI 4.8.2 20140120 (Red Hat 4.8.2-15).
- Flags de compilação:
  `-ansi -Wall -pedantic-errors -Werror -g -lm`
- Utilize comentários do tipo `/* comentário */`;
  comentários do tipo `//` serão tratados como erros pelo SuSy.

Além das observações acima, esse laboratório será avaliado pelos critérios gerais:

- Indentação de código e outras boas práticas, tais como:
    - uso de comentários (apenas quando forem relevantes);
    - código simples e fácil de entender;
    - sem duplicidade (partes que fazem a mesma coisa).

- Organização do código:
    - tipos de dados criados pelo usuário e funções bem definidas e tão independentes quanto possível.
- Corretude do programa:
    - programa correto e implementado conforme solicitado no enunciado;
    - inicialização de variáveis sempre que for necessário;
    - dentre outros critérios.