sqtpm

zap.ic.unicamp.br/sqtpm/sqtpm.cgi

Trabalho: 52-lista-generica

Linguagens: C
Data de abertura: 2016/08/29 16:00:00
Data limite para envio: 2016/10/30 18:00:00 (encerrado)
Número máximo de envios: 25
Casos-de-teste abertos: casos-de-teste.tgz

Lista Genérica

por Daniel R. Carvalho

Enunciado

Este é um trabalho extra e sem adicional de nota, fornecido apenas por questões de aprendizado, portanto deve-se dar prioridade aos trabalhos pontuados.

Deve-se implementar uma lista de funcionamento genérico, isto é, que armazene qualquer estrutura de dados sem necessidade de modificação da implementação da mesma. Para tal serão necessárias 3 etapas:

1) Definir a estrutura de dados ListElement. Uma elemento na lista deve conter um ponteiro para o conteúdo, independente do tipo de dado do mesmo, e um ponteiro para o próximo elemento.

2) Definir a estrutura de dados List. Uma lista deve conter uma sequência de ListElements, uma função de liberação de memória, uma função de comparação, e uma função de impressão (explanação abaixo).

3) Implementar as funções responsáveis pela criação de uma lista (initList), adição (ao final) de um elemento a uma lista (insertList), remoção de todas as ocorrências de um elemento (removeList), impressão dos elementos da lista (printList), e deleção de lista (freeList).

É importante notar que, devido ao fato de que o tipo de dado do conteúdo é desconhecido, não há como a lista realizar comparações, liberar memória, ou imprimir o mesmo. Por exemplo, se o conteúdo é um inteiro o free pode ser realizado diretamente (free(contents)). Contudo, se o conteúdo é uma struct contendo outras structs internas free(contents) liberará apenas a memória referente à estrutura externa, e as estruturas internas não serão liberadas. O mesmo ocorre para impressões e comparações, já que comparar inteiros é tão simples quanto a == b, porém comparar structs requer acesso aos campos internos (i.e., a.x == b.x).

Desta forma, torna-se necessário que a lista tenha acesso a funções de liberação, impressão e comparação referentes ao conteúdo que ela contém. Estas devem ser providas na criação da lista através de ponteiros de função.

Arquivos

Devem ser implementados tanto um genericList.c quanto um genericList.h que satisfaçam as condições elaboradas na Seção Enunciado. O arquivo genericList.h deve conter apenas as assinaturas dos métodos e declaração das estruturas, e genericList.c deve conter as implementações.

Caso o aluno tenha dificuldade na criação das assinaturas das funções e estruturas de dados ele pode solicitar um arquivo genericList.h para direcionamento, mas isto só deve ser feito em último caso, já que pensar em como as assinaturas e estruturas devem estar organizadas é parte do problema.

Entrada

As entradas contém um número referente ao tipo de dados a ser utilizado (vide Seção Exemplo), seguido por uma sequência de comandos. Os comandos válidos estão listados na Tabela abaixo.

Tabela de comandos que devem ser aceitos pelo programa.

Comando	Funcionalidade	Parâmetros
INSERT E ou I E	Adicionar um elemento Elemento. ao final da lista ordenada	E - Elemento
REMOVE E ou R E	Remover todas as ocorrências de um elemento, se existirem	E - Elemento
PRINT ou P	Imprimir no console o conteúdo da lista, um elemento por linha	Nenhum

E varia de acordo com o tipo de dado sendo utilizado no exemplo. Se for uma estrutura, os campos da mesma são fornecidos separados por espaços (vide exemplo 2).

Os arquivos de entrada são nomeados da seguinte forma: arqX-Y.in. X indica o exemplo, e consequente estrutura de dados que deve ser empregada (vide Seção Exemplo), e Y indica o número do exemplo.

Saída

A saída deve ser gerada de acordo com a sequência de comandos utilizados. Visando manter um padrão na impressão, estruturas complexas devem ter seu conteúdo separado por vírgulas, e cada elemento da lista deve ser impresso em uma linha.

Exemplo

Exemplos de utilização do programa utilizando estruturas de dados distintas para o conteúdo.

Exemplo 1 (arq1-1.in)

Estrutura de dados do conteúdo
int

Entrada

```
1
I 1
I 2
I 3
I 4
I 5
R 1
R 3
P
```

Saíida

```
2
4
5
```

Exemplo 2 (arq2-1.in)

Estrutura de dados do conteúdo

```
struct{
  int idade;
  char nome[20];
}
```

Entrada

```
2
I 34 Mariana
I 10 Andre
I 23 Sara
I 16 Roberto
I 52 Michael
R Mariana
R Sara
P
```

Saída

```
10, Andre
16, Roberto
52, Michael
```
