# Lista homogênea de objetos heterogêneos

**Antônio Gomes Xavier de Moura**[1], **Bruno Vercelli**[1],
**João Marcos Cucovia**[1], **Juan Marcos Braga Faria**[1], **Luiz Oliveira**[1]

[1]Instituto de Matemática e Computação – Universidade Federal de Itajubá (UNIFEI)
Caixa Postal 50 – 37.500-903 – Itajubá – MG – Brazil

antoniosames@unifei.edu.br, brunovercelli@unifei.edu.br,

joaomarcos2803@unifei.edu.br, juan.braga@unifei.edu.br,

luizraul@unifei.edu.br

## 1. Introdução

### 1.1. Do conceito de estrutura de dados e sua importância

Dado uma situação problema que peça uma solução computacional, é vital que sua implementação contemple as necessidades dos usuários da aplicação, assim como a do programador. Indissociavelmente a esta ideia, a organização das informações na arte da programação deve condizer com os graus de eficiência (em termos de memória ou computação) e facilidade que se precisa de dito programa. É neste contexto que as estruturas de dados podem ser ferramentas poderosas no processo de implementação do programa, pois cada situação pode ser representada de maneiras computacionalmente diferentes.

### 1.2. Exemplo

Considere uma pessoa que faz sua compra diária ou mensal em um supermercado. Ela se dirige ao caixa, e o(a) atendente começa a registrar os produtos. Esta lista, pertencente a uma única pessoa, naquele momento, deve não apenas conter o tipo dos produtos (provavelmente representado em um código), mas as diferentes informações especificas de cada um deles: o peso para frios do açougue; a quantidade de itens para a maioria dos demais produtos; o volume daqueles que são vendidos usando esta medida (como leite de produtores locais); a informação referente ao tamanho; entre outros.

Como pode ser notado, a lista é temporária, de tamanho indeterminado, e possui não só mais de um dado, como diferentes produtos que podem fazer uso de tipos distintos de dado.

Em uma situação como esta, o caráter temporário da informação poderia ser implementado de maneira dinâmica, não pesando no programa principal enquanto não realiza a atividade de registrar as compras; o tamanho indefinido da lista é uma oportunidade perfeita para uso de uma lista encadeada; e a heterogeneidade das informações dos diferentes produtos pede por um meio de implementar as características anteriores com diferentes informações (não seria eficiente criar estruturas que possuam todos os tipos possíveis de informação, e fazer uso apenas de uma ou duas necessárias cada vez).

Desse modo, essa situação pede por uma lista dinâmica encadeada e heterogênea.

### 1.3. Sobre o conceito de lista heterogênea

Uma lista heterogênea é uma estrutura de dado que permite a associação de diferentes tipos (e quantidades) de informação a cada um dos itens de uma lista comum, ou homogênea. Deste modo, um conjunto de produtos, por exemplo, pode ser representado por uma sequência de registros idênticos pelo programa, mas ainda conter diferentes tipos de dado em um ou mais de seus itens (o que lhe permite ser chamado de ”lista homogênea de objetos heterogêneos”). A complexidade da manipulação desses dados seria então entregue a funções, configurando um TAD (Tipo Abstrato de Dado).

### 1.4. Sobre a implementação da lista heterogênea

Na linguagem C, este tipo de dado é possível graças ao uso de referência a tipos genéricos, ou seja, com uma variável de uma lista homogênea que possa apontar para um outro dado (ou conjunto de dados) de tipo indeterminado. Na linguagem C, isto é feito com um ponteiro genérico: void*, como ilustrado pela Figura 1; que demanda uma implementação que converta o ponteiro alocado para o tipo definido.

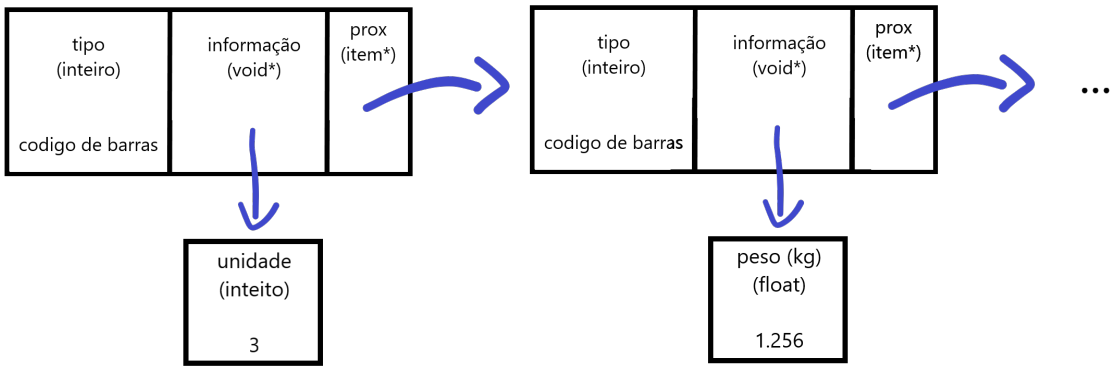


**Figura 1. Ilustração de uma lista heterogênea**

A implementação se inicia com a criação de um registro para uma lista encadeada homogênea, com alguns itens essenciais para uma estrutura do tipo (porém não exclusivamente):

a. Informação(ões) que todos os itens contenham (no exemplo usado, é o código de barras que indica o produto, e provavelmente serviria de indicação para uma tabela de preços e/ou outras informações)

b. Um meio de determinar o tipo de dado específico do item em questão (para o supermercado o código de barras serviria para a identificação, mas uma variável com códigos bem definidos também seria uma opção viável)

c. Um ponteiro genérico (do tipo void* em C) para armazenar o endereço na memória onde o(s) dado(s) estaria(m) armazenado(s).

d. Um ponteiro para o próximo item da lista (que apontara para ”NULL” para representar o último item).

Com esta estrutura homogênea em mãos, bastaria criar diferentes registros que contenham um ou mais dados específicos a cada tipo de produto, ou item da lista, e

converter o tipo do ponteiro no momento da alocação dinâmica, preferencialmente através de um TAD. Desta maneira, o programador pode abstrair o tipo de dado e apenas fazer uso da estrutura heterogênea dos itens da lista de maneira intuitiva e com mais liberdade.

## 2. Implementação

No começo do projeto foram consideradas diversas situações onde as listas heterogêneas eram úteis, como listas de compras de um supermercado, lista de animais e listas do cálculo de áreas, perímetro e volume de objetos. Dessa forma, foi decidido implementar o cálculo do volume de sólidos geométricos.

O grupo optou pela saída do programa ser na tela do usuário, conforme mostraremos abaixo:

### 2.1. Estrutura de opções do usuário

A seguir está o menu completo com as opções e subseções disponíveis ao usuário.

- Gerar lista a partir de arquivo
- Liberar lista
- Adicionar item
  - No Inicio
  - No final
- Remover item
  - Remover do inicio
  - Remover do final
  - Remover por posição
- Imprimir lista
  - Imprimir todos com volume
  - Imprimir por categoria
- Sair

### 2.2. Descrição

Uma versão bastante semelhante das operações básicas das listas encadeadas homogêneas pode ser encontrada nesta estrutura de dados. A lista heterogênea, porém, se difere da outra por conter operações que envolvam a definição do tipo de informação a ser adicionada ao respectivo local no registro da parte homogênea da lista e a necessidade de liberar a informação contida dentro da lista homogênea na função de liberar a lista.

#### 2.2.1. Sobre a geração da lista a partir do arquivo

Para a realização de uma lista gerada de um arquivo é necessário ler um arquivo de texto e alocá-lo na lista. Esse arquivo de texto irá conter informações sobre o tipo de sólido geométrico, bem como os valores necessários para se calcular o volume. Na função, primeiramente é preciso abrir o arquivo de texto, caso não exista um arquivo de texto ou caso dê erro durante a abertura do arquivo a função irá retornar um texto na tela informando o usuário. Caso contrário será feita uma leitura no arquivo com uma funcao que extrai o conteudo linha por linha (string) e o copia para um local temporario no programa (um vetor de caracteres); até que se chegue ao seu fim.

### 2.2.2. Entrada de dados do programa

A struct elemento possui os seguintes dados: tipo(inteiro), um ponteiro para as informações e uma outra struct que aponta para o próximo elemento. O que o difere das listas homogêneas comuns é que o dado ”tipo” será necessário para devida leitura dos dados da lista heterogênea, pois é através dele que o programa identificará qual registro ele deverá se basear para fazer uso das informações.

Foram definidas 5 structs de formas geométricas diferentes, onde cada forma representa dados diferentes. Por exemplo: a struct cilindro contém os dados para o raio e a altura de um cilindro, já a struct esfera contém apenas um dado do raio da esfera. A Figura 2 exibe a estrutura dos dados no arquivo que o programa deve receber.

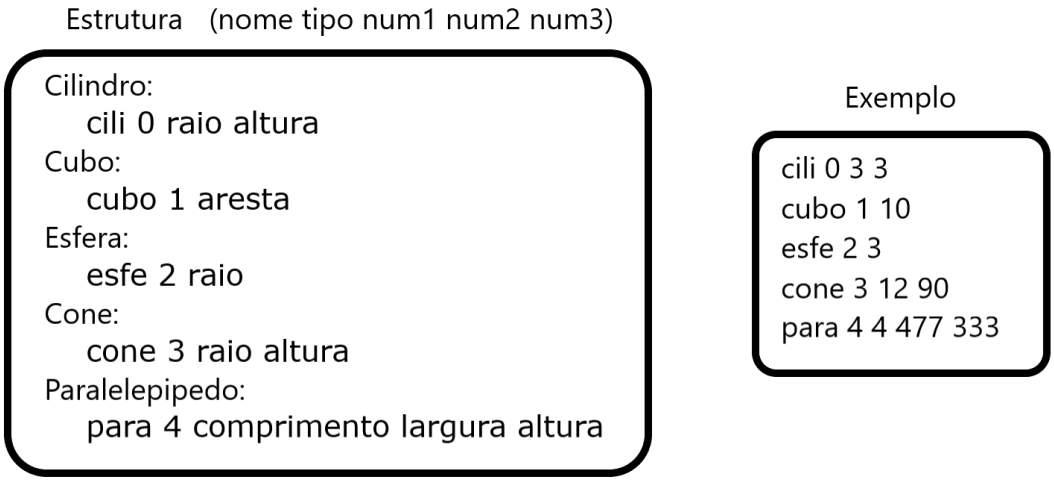


**Figura 2. Estrutura de dados de entrada do programa.**

## 3. Resultados

Ao longo do projeto foi utilizada a leitura de dados compostos a partir de um arquivo. Foi necessário realizar uma pesquisa em relação a estrutura que o arquivo iria ser utilizado, de forma que fosse possível ler diferentes tipos de dados e transformá-los em uma lista heterogênea.

Além disso, foi utilizada a manipulação de listas heterôgeneas usando TAD, onde é necessário utilizar diferentes structs (registros) com diferentes campos de dados e uma struct homogênea para armazenar essas informações das structs heterogêneas.

Ao final do projeto foram implementadas todas as funções necessárias para criar uma lista heterogênea básica, como inserir, remover, imprimir e liberar, de forma que o código da lista heterogênea está funcionando corretamente, modularizado e comentado.

## 4. Conclusão

A implementação da lista heterogênea provou ser algo custoso, em especial pela constante verificação de tipos e conversões de ponteiro. Uma vez que ela apresenta um certo custo computacional e não oferece muitas vantagens (ao menos em relação às poucas aplicações

que o grupo acredita que a demandavam), é preciso muita consideração ao se escolher a estrutura de dados heterogênea.

Adicionalmente, há uma evidente escassez de conteúdos na literatura sobre o tema, a rigor, lista homogênea de objetos heterogêneos na linguagem de programação C. Isso se deve pela sua complexidade de implementação e aplicabilidade relativamente baixa quando comparada com outras estruturas, além da natural simplicidade (e portanto distanciamento da intuição humana) da linguagem C também ser um fator de peso.

Em linguagens de nível mais alto, como Python e Java, onde se prioriza a intuição do programador e se permite um grau muito mais alto de flexibilidade e abstração; a estrutura se faz muito mais presente na programação, uma vez que há uma fundação mais solida e complexa para suportar toda a conveniência oferecida.

Apesar de todos esses fatores, é evidente que a lista se provaria útil em certos casos, e que, portanto, um programador bem preparado com conhecimento desta e outras estruturas terá a aptidão para produzir um código capaz de representar e manipular os dados da aplicação, de maneira mais eficiente e condizente com as necessidades dele e dos usuários finais.

## 5. Referências Bibliográficas

CodeVault. (2020, July 7). Read an array of structs in C [Video]. Youtube. https://www.youtube.com/watch?v=shYMgRcjm5A

W. Celes e J. L. Rangel, INTRODUÇÃO À LINGUAGEM C, Capitulo 10, Versão 2.0, CENTRO DE COMPUTAÇÃO, UNICAMP. Disponivel em: https://www.ic.unicamp.br/%7Era069320/PED/MC102/1s2008/Apostilas/Cap10.pdf

## 6. Anexos

Segue abaixo o link do repl.it com o código e o link do vídeo da nossa implementação:

Link do Repl.it:

Link do video: