**SMART POINTERS**

GRUPO 07

Mateus Pim Santos

Karlos Helton Braga

Victor Gomes Sampaio

Max Lucio Martins de Assis

# Sumário

[**Sumário**](#_555a26mfn9yw) **2**

[**Problema**](#_h6mgnv3rrjvi) **3**

[**Solução**](#_c7ch8gj8j33a) **4**

[**Código**](#_j7pp5uv0yud) **5**

# Problema

*Explique a utilidade dos ponteiros e os problemas de sua má utilização, convergindo para os smart pointers. Faça corretamente as citações de fontes consultadas. Trabalhos sem referência de fontes na Introdução serão desconsiderados;*

Ponteiro, em programação, é um tipo de dado cujo valor se refere diretamente a um outro valor alocado em outra área da memória, através de seu endereço[[1]](#footnote-0). Ou seja, o ponteiro contém o endereço de outra variável, função ou objeto na memória.

(Aqui vai um parágrafo explicando a utilidade dos ponteiros).

Richard Reese, em seu livro *‘Understanding and Using C Pointers’,* cita alguns exemplos de problemas da má utilização dos ponteiros:

* Acessar arrays e outras estruturas de dados além de seus limites.
* Referenciar variáveis após elas terem deixado de existir.
* Referenciando memória alocada de fila de prioridade após que ela foi liberada.
* Referencing heap allocated memory after it has been released.
* Desreferenciar um ponteiro antes que a memória tenha sido alocada para ele.

Quando mal utilizados, geram grandes problemas na execução de código pois o problema independe do ponteiro estar correto ou não, mas sim a informação apontada por ele, quando se utiliza um ponteiro ruim, você está lendo lendo “lixo” da memória e se for escrever poderia estar sobrescrevendo alguma informação de outra parte da execução do código[[2]](#footnote-1).

Tendo em vista, que ponteiros são grandes ferramentas porém são altamente voláteis em sua execução fora criado os smart pointers para tentar auxiliar aos programadores a cometer menos erros quando se utilizam de ponteiros em seu código. Smart pointers, por sua vez, são classes modeladas para gerenciar memória dinamicamente alocada, ela fornece todas as interfaces existentes por ponteiros “normais” com algumas exceções, por exemplo: durante a construção ele cria espaço na memória e desaloca a mesma quando o ponteiro sai do escopo da função/método, desta maneira o programador não necessita, obrigatoriamente, de se preocupar em desocupar o espaço de memória depois de utilizada os ponteiros[[3]](#footnote-2).

# Solução

*Explique o que são e como funcionam smart pointers em C++ 11. Defina unique\_ptr & shared\_ptr;*

Com a integração dos Smart Pointers[[4]](#footnote-3) foram apresentados os unique\_ptr e shared\_ptr:

* Unique\_ptr procura ter a ideia de ponteiros únicos, assim seu objetivo é ter um único ponteiro para um certo objeto em qualquer tempo da execução e desalocar memória quando o ponteiro sai do escopo.
* Shared\_ptr procura ter a ideia de ponteiros compartilhados, assim seu objetivo é ter múltiplos ponteiros, sendo possível passar por referência o ponteiro para outras funções, se referindo a um objeto e quando o último ponteiro compartilhado sair do escopo (que está inserido) a memória é desalocada automaticamente.

# Código

Exemplo I: FAÇA um código C++ do grupo mostrando o uso do unique\_ptr; não copie da Internet ou das fontes consultadas - desenvolva o SEU exemplo;

Exemplo II: FAÇA um código C++ do grupo mostrando o uso do shared\_ptr; não copie da Internet ou das fontes consultadas - desenvolva o SEU exemplo;

Considerações Finais

1. Ponteiro (programação): https://pt.wikipedia.org/wiki/Ponteiro\_(programa%C3%A7%C3%A3o) [↑](#footnote-ref-0)
2. What are the major problems with pointers in C/C++ programming?: https://www.quora.com/What-are-the-major-problems-with-pointers-in-C-C++-programming [↑](#footnote-ref-1)
3. C++11 Smart Pointers: https://www.codeproject.com/Articles/541067/Cplusplus-Smart-Pointers [↑](#footnote-ref-2)
4. Smart Pointers: https://en.cppreference.com/book/intro/smart\_pointers [↑](#footnote-ref-3)