Nome: Éliton Pereira Melo
Disciplina: POO
matrícula: 536157

Escolha duas linguagens além da sua linguagem de estudo e tente comparar como funciona o processo de alocação e desalocação de memória.

Pesquise sobre:

- Alocação é explícita ou implícita? Sempre new para criar novos objetos?

- Permite desalocação manual ou é possível forçar a coleta de lixo?

- Se não tiver garbage colector, como funciona a desalocação?

- Faz contagem de referências para saber a hora de liberar as variáveis?

- Como funcionam vazamentos de memória nessas linguagens? Ou ela não tem vazamento de memória?

Linguagem: C

- Alocação é explícita ou implícita? Sempre new para criar novos objetos?

Em C, a alocação é explícita, ou seja, o programador precisa solicitar explicitamente a memória necessária para os dados. Isso pode ser feito usando as funções como ***malloc()*** e ***calloc()*** para alocar a memória dinamicamente. Não, em C não se usa o operador ***new*** para alocar memória dinamicamente. Em C, para alocar memória dinamicamente, utiliza-se a função ***malloc***, que é declarada na biblioteca padrão ***"stdlib.h"***. Para criar um novo objeto em C, é necessário alocar a memória para este objeto com ***malloc*** e depois inicializá-lo através de funções de inicialização ou atribuição de valores.

- Permite desalocação manual ou é possível forçar a coleta de lixo?

Em C, a alocação de memória é feita explicitamente usando funções como ***malloc***, ***calloc*** e ***realloc***, e a desalocação é feita usando a função ***free***. Não há coleta de lixo automática em C, portanto a desalocação de memória deve ser feita manualmente pelo programador.

- Se não tiver garbage colector, como funciona a desalocação?

Em C, a desalocação da memória alocada dinamicamente é feita manualmente pelo programador usando as funções ***free()*** ou ***realloc()***. O programador precisa ter o controle sobre quando a memória alocada deve ser liberada, pois o sistema não tem um garbage collector para fazer isso automaticamente.Em resumo, em C a alocação e

desalocação de memória é totalmente manual, sem o auxílio de um garbage collector ou contagem de referências.

- Faz contagem de referências para saber a hora de liberar as variáveis?

Não, a linguagem C não possui um mecanismo de coleta de lixo ou contagem de referências embutido. A alocação e liberação de memória é feita manualmente pelo programador usando as funções ***malloc()*** e ***free()*** ou ***calloc()*** e ***realloc()***. É responsabilidade do programador garantir que toda memória alocada seja liberada corretamente quando não for mais necessária para evitar vazamentos de memória e falhas no programa. Por isso, é importante que o programador tenha um bom entendimento do gerenciamento de memória em C para escrever código seguro e eficiente.

- Como funcionam vazamentos de memória nessas linguagens? Ou ela não tem vazamento de memória?

Em C, é possível ocorrer vazamento de memória caso o programador não desaloque manualmente a memória alocada dinamicamente para variáveis usando ***malloc*** ou ***calloc***. Isso acontece quando o programa aloca memória dinamicamente para uma variável e não a libera após o término do seu uso, o que pode levar a um aumento constante do consumo de memória do programa e possivelmente causar falhas no sistema operacional quando a memória disponível é esgotada. No entanto, ao contrário de algumas linguagens modernas, C não possui um mecanismo automático de coleta de lixo ou gerenciamento de memória para detectar e recuperar automaticamente a memória não utilizada. Portanto, a responsabilidade de gerenciar a alocação e desalocação de memória é deixada para o programador em C. Isso significa que é importante que os programadores sejam cuidadosos ao usar funções que alocam dinamicamente a memória, e que sempre liberem a memória alocada quando ela não for mais necessária.

Linguagem: C++

- Alocação é explícita ou implícita? Sempre new para criar novos objetos?

Já em C++, a alocação pode ser tanto explícita quanto implícita, pois a linguagem fornece suporte tanto para alocação dinâmica quanto para alocação estática de memória. Em relação à criação de novos objetos em C++, não é necessário sempre usar o operador ***new***. Existem outras formas de criar objetos em C++, como a alocação automática em stack e a alocação estática em tempo de compilação.

- Permite desalocação manual ou é possível forçar a coleta de lixo?

Em C++, a desalocação é feita usando o operador ***delete***. Também é possível usar ***new[]*** para alocar matrizes de objetos e ***delete[]*** para deslocá-las. A partir do C++11, a linguagem adicionou suporte a alocação de memória automática com o uso de ***std::shared_ptr*** e ***std::unique_ptr***, que gerencia a desalocação automaticamente quando o objeto de ponteiro inteligente sai do escopo.

- Se não tiver garbage colector, como funciona a desalocação?

No C++, a desalocação de memória é feita manualmente pelo programador utilizando o operador ***delete*** para desalocar um objeto ou ***delete[]*** para desalocar um array de objetos. O operador ***delete*** libera a memória alocada pelo operador ***new*** e chama o destrutor do objeto antes de desalocá-lo. O programador deve ser responsável por chamar os operadores ***delete*** e ***delete[]*** no momento apropriado para evitar vazamentos de memória e erros de acesso à memória não alocada.

- Faz contagem de referências para saber a hora de liberar as variáveis?

Sim, em C++ é comum o uso de contagem de referências como técnica de gerenciamento de memória para liberar variáveis quando elas não são mais necessárias. Nessa abordagem, cada objeto possui uma contagem de referências, que é incrementada quando uma nova referência para o objeto é criada e decrementada quando uma referência é destruída. Quando a contagem de referências chega a zero, isso indica que não há mais nenhuma referência para o objeto e ele pode ser liberado da memória. No entanto, essa não é a única técnica de gerenciamento de memória disponível em C++. É possível usar outras técnicas, como a alocação estática, alocação automática, alocação dinâmica com uso de ponteiros e a desalocação manual com o operador ***delete***.

- Como funcionam vazamentos de memória nessas linguagens? Ou ela não tem vazamento de memória?

Assim como em C, o C++ também pode ter vazamento de memória se o programador não gerenciar corretamente a alocação e desalocação de memória. O C++ possui recursos de gerenciamento de memória, como a alocação de memória estática, automática e dinâmica. A alocação estática de memória é feita em tempo de compilação e é liberada automaticamente quando o programa é encerrado. A alocação automática é feita em tempo de execução, por meio da pilha de chamadas de função, e é liberada automaticamente quando a função que alocou a memória retorna. Já a alocação dinâmica é feita por meio de operadores como ***new*** e ***delete***, e cabe ao programador gerenciar manualmente a alocação e desalocação de memória. Caso a memória não seja desalocada adequadamente, ocorrerá um vazamento de memória. Para ajudar a evitar vazamentos de memória, o C++ possui recursos como ponteiros inteligentes (smart pointers) que gerenciam automaticamente a alocação e desalocação de memória.