Segmentação da Memória

↳ A memória RAM é dividida em diversos segmentos, dentre eles a Stack e a Heap

Quais as principais diferenças entre a Stack e a Heap?

* Stack:
  ↳ Uma forma otimizada para organizar dados na memória
  ↳ Armazena variáveis temporárias criadas por uma função
  ↳ Quando a main() é finalizada, a memória da variável será apagada

* Heap:
  ↳ É a organização de memória mais flexível que permite o uso de qualquer área lógica disponível
  ↳ É onde os programas armazenam variáveis globais
  ↳ Mais complexa de rastrear quais partes da Heap estão alocadas ou livres

Parameter	Stack	Heap
Type of data structures	A stack is a linear data structure.	Heap is a hierarchical data structure.
Access speed	High-speed access	Slower compared to stack
Space management	Space managed efficiently by OS so memory will never become fragmented.	Heap Space not used as efficiently. Memory can become fragmented as blocks of memory first allocated and then freed.
Access	Local variables only	It allows you to access variables globally.
Limit of space size	Limit on stack size dependent on OS.	Does not have a specific limit on memory size.
Resize	Variables cannot be resized	Variables can be resized.
Memory Allocation	Memory is allocated in a contiguous block.	Memory is allocated in any random order.
Allocation and Deallocation	Automatically done by compiler instructions.	It is manually done by the programmer.
Deallocation	Does not require to de-allocate variables.	Explicit de-allocation is needed.
Cost	Less	More
Implementation	A stack can be implemented in 3 ways simple array based, using dynamic memory, and Linked list based.	Heap can be implemented using array and trees.
Main Issue	Shortage of memory	Memory fragmentation
Locality of reference	Automatic compile time instructions.	Adequate
Flexibility	Fixed size	Resizing is possible
Access time	Faster	Slower

Como é organizado o armazenamento na Stack? O armazenamento na Heap é feito da mesma forma?

* Stack:
  - ↳ Estrutura Last in First Out (LiFo)
    - ↳ O bloco de memória mais recente a ser alocado é sempre o próximo blo a ser desalocado
  - ↳ É uma estrutura de pilha
  - ↳ Porção contígua de memória reservada para empilhar os dados necessários durante a execução de blocos de código
  - ↳ Cada necessidade de alocação é um trecho da stack que é usado por um ponteiro que se "movimenta" para indicar que uma nova parte é reservada
  - ↳ Quando o pedaço reservado não for mais necessário, este marcador se movimenta "para trás", indicando que esses espaços podem ser sobrepostos

* Heap:
  - ↳ Sua alocação é bastante flexível
  - ↳ Esta alocação pode ocorrer fisicamente em qualquer parte livre da memória
  - ↳ Alocação mais dinâmica

As duas segmentações da memória diferem muito em tamanho (ou capacidade de armazenamento)?

- ↳ O tamanho da stack é definido quando uma thread é criada, já na Heap, seu tamanho é definido quando a aplicação se inicia, mas pode crescer a medida que mais espaço for necessário
- ↳ O tamanho da stack é bem menor que o da Heap, e imutável, enquanto que a Heap tem basicamente toda a parte da memória RAM que está vazia

Em que situações é aconselhável que utilizemos a memória Stack? E a memória Heap?

* Stack
  - ↳ Te ajuda a organizar os dados em um método LIFO
  - ↳ Assim que você declara uma variável local, ela fica na Stack. É automaticamente destruída depois do retorno da função
  - ↳ Limpa os objetos automaticamente

* Heap:
  - ↳ Acessar variáveis globais
  - ↳ Maior flexibilidade, sem limite de memória
  - ↳ Previne o Stack Over Flow

Existem prejuízos em abusar da memória Stack?

↳ Devido a baixa capacidade de memória da Stack, ao alocar um espaço muito grande de memória ela pode "estourar", ou seja, irá dar um erro por uso de memória maior que ela podia lidar
↳ Esse erro é conhecido como Stack Over Flow