Lógica de Programação

Tipos de Dados e Variáveis

Tipos de Dados

- **Inteiros:** Representam números sem casas decimais, podendo ser positivos ou negativos.
  Exemplos: 10, 47, 3490, -2000, -39845.
- **Reais (Ponto Flutuante):** São números com casas decimais, também podendo ser positivos ou negativos.
  Exemplos: 11.2, 604.29, -239.81, 7.89213, 3, 7.
- **Caracteres:** Representam todos os símbolos presentes na tabela ASCII, incluindo letras, números e caracteres especiais.
  Exemplos: 'a', 'A', '&', '$', '*', "Vitor", "21", "856012780".
  Observação: Um único dígito como '1' é considerado um caractere, enquanto uma sequência como "5898" é uma cadeia de caracteres (string).
- **Booleano:** Representa um valor lógico, podendo ser verdadeiro (true) ou falso (false).

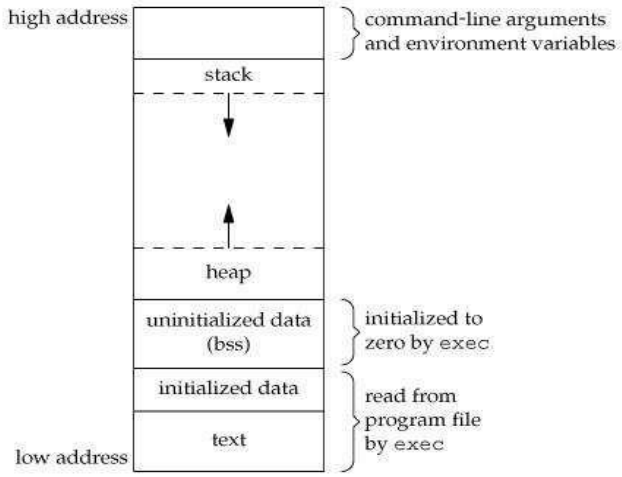
Variáveis

Em programação, variáveis são elementos fundamentais para armazenar dados temporariamente na memória do computador, permitindo que os valores sejam utilizados e manipulados ao longo do código. Estão sempre associadas a algum dado do problema que se busca resolver.

No que diz respeito à alocação de memória, diferentes segmentos de

memória são utilizados para organizar e gerenciar os dados. Estes segmentos

são cruciais para o bom funcionamento de um programa.



High Address* e *Low Address

A memória de um computador é organizada como uma sequência de endereços numerados, que vão de um valor baixo (*low address*) até um valor alto (*high address*). Cada endereço representa uma posição na memória, onde dados e instruções são armazenados.

- ***Low Address*:** Refere-se ao início ou à parte mais baixa do espaço de memória. Por exemplo, se a memória vai de 0 a 1.000.000, os *low addresses* seriam os mais próximos de 0. Instruções de código, como as da *text segmen*t, são geralmente armazenadas nessa região.
- ***High Address*:** Refere-se ao final ou à parte mais alta da memória, como algo próximo de 1.000.000 no exemplo anterior. Nesta região, são armazenados dados alocados dinamicamente, como os do *heap*. O *stack* também pode crescer em direção aos *high addresses*, dependendo da arquitetura.

Stack

A *stack* (ou "pilha") é uma área da memória usada para armazenar variáveis locais e informações de controle durante a execução de uma função. A memória da *stack* é gerenciada automaticamente pelo sistema operacional, que cuida da alocação e desalocação conforme as funções são chamadas e encerradas.

Heap

O *heap* é a região da memória destinada à alocação dinâmica de dados. Quando, por exemplo, a função *malloc* (o nome é abreviatura de *memory allocation*) em C ou

new em C++ é utilizada, os dados são alocados no *heap*. A diferença em relação à *stack* é que a memória do *heap* pode perdurar por toda a execução do programa, até que seja liberada explicitamente por *free* ou *delete*.

Alocação dinâmica de memória. Como redimensionar memória? (usp.br)

BSS (*Uninitialized Data Segment*)

A BSS (*Block Started by Symbol*) é uma seção da memória reservada para variáveis globais ou estáticas não inicializadas (ou inicializadas com zero). Apesar de ser uma parte importante da estrutura de memória, não ocupa espaço no arquivo binário do programa, já que os valores dessas variáveis são atribuídos apenas durante a execução.

Initialized Data Segment

A *Initialized Data Segment* armazena variáveis globais ou estáticas que foram explicitamente inicializadas com algum valor. Essa seção contém os dados que já estão prontos para uso no início da execução do programa.

Text Segment

A *text segment* é a parte da memória onde o código executável do programa é armazenado. Ela contém as instruções de máquina que o processador executa. Para garantir a integridade do código, essa área costuma ser marcada como somente leitura.

Resumo

- ***High Address:*** Parte superior do espaço de memória.
- ***Low Address:*** Parte inferior do espaço de memória.
- ***Stack:*** Aloca variáveis locais e controla a execução das funções (alocação automática).
- ***Heap:*** Utilizado para alocação dinâmica de memória (alocação manual).
- ***BSS (Uninitialized Data):*** Armazena variáveis globais ou estáticas não inicializadas.

- ***Initialized Data:*** Armazena variáveis globais ou estáticas inicializadas explicitamente.
- **Text Segment: Armazena o código executável.**