**Alocação dinâmica de memória**

Tal como a stack, a heap expande e contrai dinamicamente em tempo de execução como resultado da alocação ou libertação de memória.

A heap aumenta em direção e endereços mais altos.

- O programador não conhece a quantidade de memória necessária para lidar com os dados:

- tem de ser pessimista quando efetua a alocação estática.

- aloca memória suficiente para o pior caso possível -> a memória é usada ineficientemente

- Estruturas de dados complexas: listas e arvores

**ALOCAÇÃO DINÂMICA DE MEMORIA EM C**

**Void\* malloc(size\_t size);**

Aloca um bloco de memória continuo de pelo menos size bytes e devolve um apontador para esse bloco

A memória não é inicializada

Se o malloc() encontrar um problema (o programa requisita um bloco de memoria maior que a memoria virtual disponível) retorna null

**Void free(void \*ptr);**

Liberta um bloco de memoria da heap alocado previamente com uma função de alocação

O argumento ptr deve apontar para o inicio de um bloco alocado obtido através de uma função de alocação de memoria

**EXEMPLO DE malloc() e free()**

#include <stdlib.h> //maloc() e free() fazem parte da stdlib

/\*…\*/

Int \*ptr\_int = null; //declarar um apontador

Ptr\_int = (int\*)malloc(10\*sizeof(int)); //alocar espaço para 10 int na heap

//AS DUAS LINHAS ACIMA PODEM SER FUNDIDAS NUMA SÓ E FAZ MAIS SENTIDO DO QUE COMEÇAR POR INICIALIZAR COM NULL

Ptr\_int[0] = 10; //array de 10 inteiros

\*(ptr\_int+1)=20;

Free(ptr\_int); //libertar espaço de memoria alocado

/\*…\*/

**Void\* calloc(size\_t n, size\_t size);**

Aloca um bloco de memoria continuo de pelo menos n\*size bytes

Todos os bytes do cloco de memoria são inicializados com zero

Devolve um apontador para esse bloco, ou null em caso de erro

**EXEMPLO DO calloc();**

Semelhante, mas:

…

Ptr\_int = (int\*)calloc(100, sizeof(int));

…

**Void\* realloc(void\* ptr, size\_t size);**

Modifica o tamanho de um bloco de memoria previamente alocado com malloc() ou calloc()

Os dados serão mantidos se o tamanho aumentar, serão truncados se o tamanho diminuir

No cado do tamanho aumentar, o bloco de memoria adicional não é inicializado

Devolve um apontados para esse bloco, ou null em caso de erro

**EXEMPLO DO realloc()**

Igual aos de cima, mas:

…

Ptr\_int = (int\*)calloc(100,sizeof(int)); //alocar espaço para 100 int na heap

Ptr\_int=(int\*)realloc(ptr\_int, 101\*sizeof(int)); //alocar espaço para mais 1 int na heap

…

Se o realloc() falhar, o apontador para a zona de memória é perdido.

Solução: usa-se um apontador temporário e confirma-se o retorno do realloc()

* ESTA SITUAÇAO É DENOMINADA DE MEMORY FREGMENTATION

**EXEMPLO DE CONFIRMACAO DO RETORNO DO REALLOC()**

…

Int \*ptr\_int = null, \*ptr\_tmp=null; //declarer e inicializar os apontadores

Const int N = 100; //quantidade de ints para os quais se pretende alocar a memoria

Ptr\_int = (int\*)calloc(n\*sizeof(int));

//ALOCAR ESPAÇO PARA MAIS 1 INT NA HEAP

//NOTA:O RETORNO É ARMAZENADO NUM APONTADOR TEMPORARIO

Ptr\_tmp=(int\*)realloc(ptr\_int,(n+1)\*sizeof(int));

If(ptr\_tmp!=null){ //se o realloc tiver sucesso aceitar o novo bloco de memoria

Ptr\_int = ptr\_tmp;

Ptr\_tmp=null;

}

Free(ptr\_int);

…

VALGRIND -> PROGRAMA DA LINHA DE COMANDOS (não é preciso instalar na maquina virtual, já deve estar lá)

Valgrind ./nome\_executavel -> DÁ UM RELATORIO DO QUE SE ESTÁ A PASSAR, RELATIVAMENTE AOS BLOCOS DE MEMORIA E OUTRAS ASNEIRAS

RESUMO

Podemos gerir a memoria de forma dinâmica criando blocos de memoria na heap à medida das necessidades

Na alocação dinâmica de memoria, a memoria é alocada em tempo de execução

A alocação dinâmica de memoria permite manipular estruturas de dados cujo tamanho passa a ser flexível e que pode ser modificado em qualquer altura do programa

É necessário quando não se faz ideia do tamanho que a estrutura de dados está a ocupar