UNIVERSIDADE FEDERAL DE ITAJUBÁ

Algoritmos e Estrutura de Dados I

CTCO-01

Alocação Dinâmica

Vanessa Cristina Oliveira de Souza

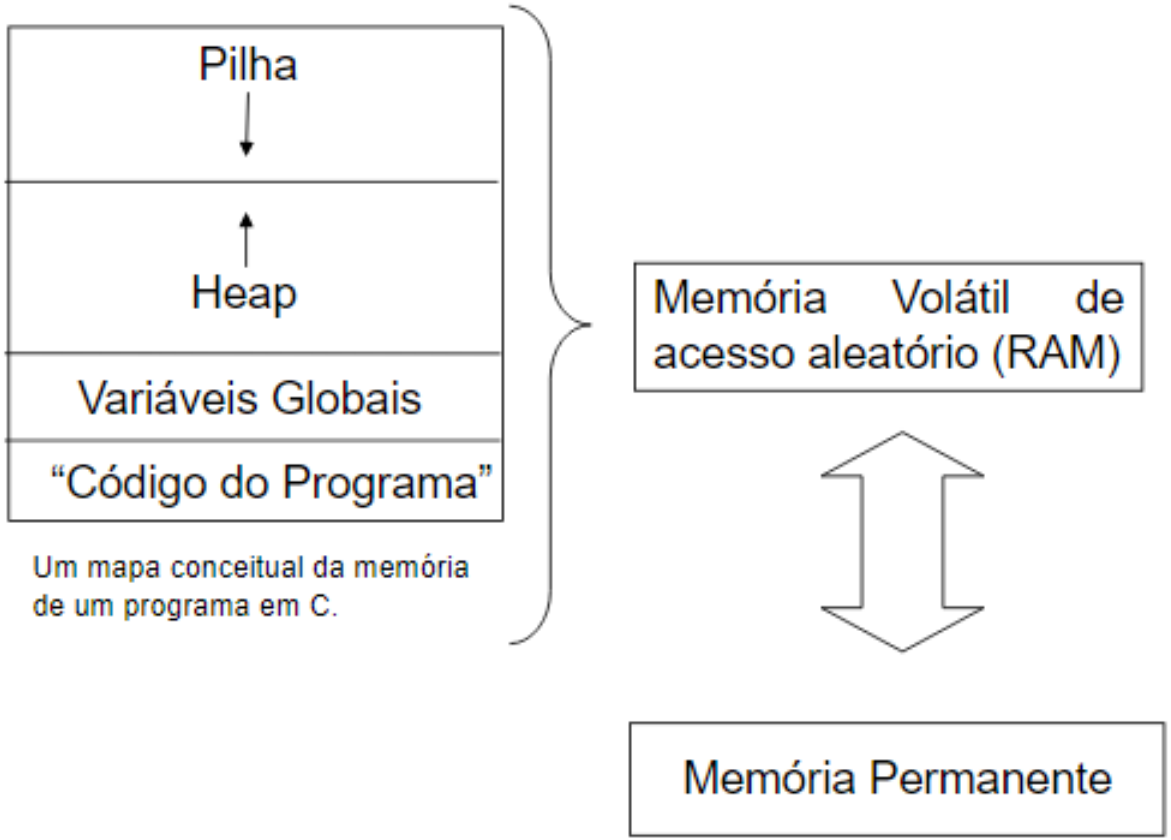
Introdução

- **Variável** é um local reservado na memória para armazenar um tipo de dado.

- Ao declarar uma variável, o programador está ALOCANDO MEMÓRIA!
  - Ao declarar uma variável, o **COMPILADOR** reserva uma região na memória para ela.
  - A quantidade de memória depende do tamanho do tipo de dado para o qual a variável foi declarada.

Introdução

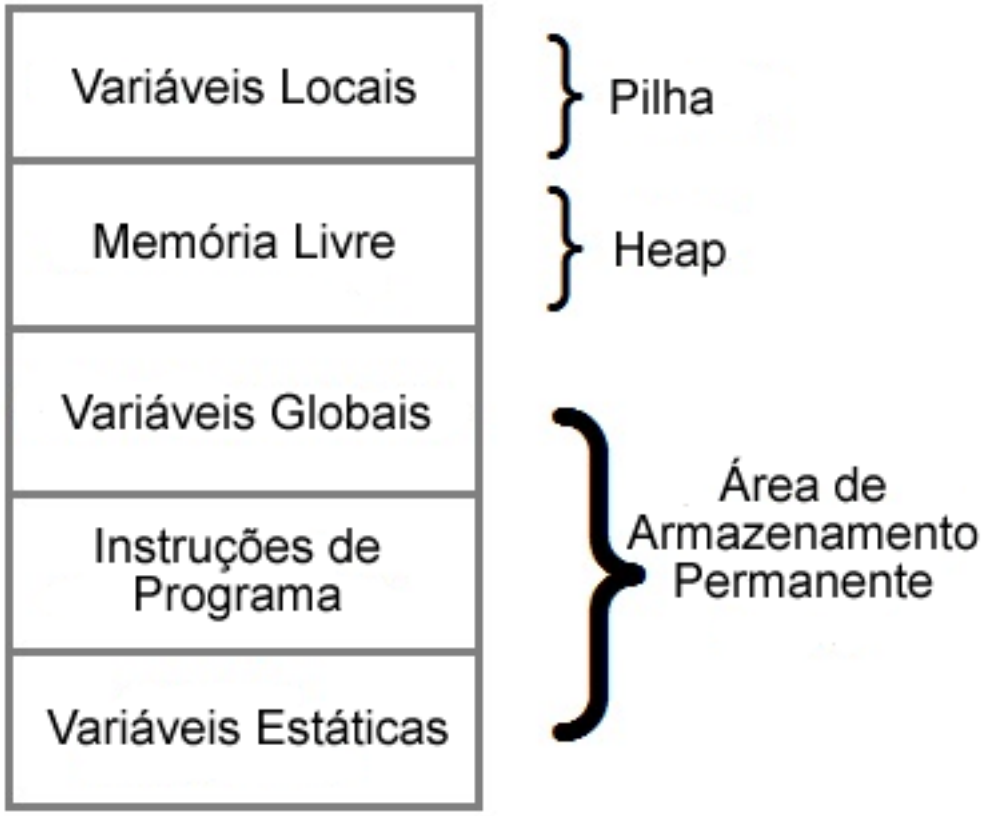
- Mapa de memória em C



Um mapa conceitual da memória de um programa em C.

Introdução

- Mapa de memória em C
  - Variável Global
    - Alocada em tempo de compilação
  - Variável Estática
    - Alocada em tempo de compilação
  - Variável Local
    - Alocada usando a pilha de execução do programa



Introdução

- Veja os exemplos abaixo:
  - int i;
  - float k;
  - char c ;
  - int vet[30];

- Os códigos acima são declarações de variáveis
  - ALOCAM MEMÓRIA ESTATICAMENTE
    - OU SEJA, ANTES QUE O PROGRAMA COMECE SER EXECUTADO
    - No caso do vetor, a alocação estática ‘enxerga’ a variável como um único bloco, que ocupa posições contíguas na memória.

Alocação de Variáveis em C

- Variáveis e estruturas de dados como vetor e matriz podem ser alocadas de forma estática ou dinâmica.

- A alocação estática acontece quando no momento da compilação, a quantidade de memória necessária é conhecida.

- E quando não se sabe o quanto de memória vamos utilizar?

Alocação Dinâmica vs Alocação Estática

- Quando o próprio programador define a quantidade de memória necessária para uma variável, especificando o tipo e tamanho, a alocação de memória é feita pelo compilador e realizada antes que o programa esteja sendo executado.
  - ALOCAÇÃO ESTÁTICA

- Quando não for possível definir de antemão a quantidade de memória necessária para armazenar o que se deseja, a alocação vai acontecer enquanto o programa estiver sendo executado.
  - ALOCAÇÃO DINÂMICA

Alocação Dinâmica de Memória

Alocação Dinâmica

- Alocação dinâmica é o meio pelo qual um programa pode obter memória enquanto está em execução.
  - Haverá momentos em que um programa precisará usar quantidades de armazenamento variáveis.
  - Ou seja, a quantidade de memória a alocar só se torna conhecida durante a *execução* do programa.
  - Para lidar com essa situação é preciso recorrer à alocação *dinâmica* de memória.

Alocação Dinâmica

- Exemplo:
  - *Ler um conjunto de dados das turmas do curso de ciência da computação a partir de um arquivo e gerar um relatório sobre o índice de aprovação de cada turma.*

  - As turmas possuem quantidade de alunos variável.
  - Seu programa deve funcionar para qualquer tamanho de turma.
  - Como definir o tamanho da estrutura sem antes ler o arquivo?

Alocação Dinâmica

- Exemplo:
  - *Ler um conjunto de dados das turmas do curso de ciência da computação a partir de um arquivo e gerar um relatório sobre o índice de aprovação de cada turma.*
  - *void carregaDados (char *nomeArquivo, float *notas)*
    - A cada execução do programa, o tamanho do vetor de notas é diferente e depende da quantidade de alunos da turma, que será lida do arquivo, em tempo de execução.
    - Portanto, o ideal é alocar esse vetor em tempo de execução, depois de conhecer o tamanho da turma.

Alocação Dinâmica vs Alocação Estática

- Exemplo:
  - *void carregaDados (char *nomeArquivo, float *notas)*
    - O nome de um vetor é um ponteiro
    - **notas* (valor apontado por notas) é equivalente a notas[0]
    - **(notas + i)* é equivalente a notas[i]
    - A única diferença entre declarar o vetor notas como um ponteiro para float *(float *notas)* ou como um vetor de elementos do tipo float *(float notas[10])* está na **alocação de memória**.

Subsistema de Alocação Dinâmica

- A linguagem C oferece um subsistema para alocação dinâmica, cujas principais funções são:
  - calloc
  - malloc

  Aloca memória
  - free

  Libera memória
  - realloc

  Realoca a quantidade de memória alocada, para mais ou para menos

As funções pertencem a biblioteca stdlib.h

Subsistema de Alocação Dinâmica

- As funções de alocação de memória alocam a quantidade de **bytes** definida pelo programador e retornam um **ponteiro** para o início da memória alocada.
  - Se não for possível alocar a memória, as funções retornam NULL
- O endereço retornado pelas funções são do tipo *void* e, por isso, faz-se necessário realizar um *cast* para o tipo desejado.
- O espaço alocado dinamicamente permanece reservado até que explicitamente seja liberado pelo programa.
  - Se o programa não liberar um espaço alocado, este será automaticamente liberado quando a execução do programa terminar.

Subsistema de Alocação Dinâmica

- Função malloc
  - *void *malloc(size_t size);*
  - Aloca uma quantidade de memória igual a *size* bytes

```
char *p;
p = (char *) malloc(1000);
```

  - Após a atribuição, *p* aponta para o primeiro dos 1000 bytes de memória livre.

Subsistema de Alocação Dinâmica

- Função malloc
  - *void *malloc(size_t size);*

```
int *p;
p = (int *) malloc(50*sizeof(int));
```

    - Aloca espaço para 50 inteiros

- A função *sizeof* garante portabilidade
  - Compilador com int de 2 bytes
  - Compilador com int de 4 bytes

Subsistema de Alocação Dinâmica

- Função calloc
  - *void *calloc(size_t num, size_t size);*
  - Aloca uma quantidade de memória igual a *num*size* bytes

```
float *p;
p = (float *) calloc(50, sizeof(float));
```

  - Aloca espaço para 50 inteiros

Subsistema de Alocação Dinâmica

- As funções *malloc* e *calloc* retornam NULL caso não consigam alocar a memória por, por exemplo, não haver espaço no Heap.

- Por isso é importante validar a alocação antes de usar o ponteiro.

```
float *p;
p = (float *) calloc(50, sizeof(float));

if (!p)
{
    printf("Erro ao alocar a memória");
    exit(1);
}
```

Subsistema de Alocação Dinâmica

- As funções *malloc* e *calloc* retornam NULL caso não consigam alocar a memória por, por exemplo, não haver espaço na Heap.

- Por isso é importante validar a alocação antes de usar o ponteiro.

```
float *p;
p = (float *) malloc(50*sizeof(float));

if (p == NULL)
{
    printf("Erro ao alocar a memória");
    exit(1);
}
```

Subsistema de Alocação Dinâmica

- Os códigos são equivalentes!

```c
float *p;
p = (float *) malloc(50*sizeof(float));

if (p == NULL)
{
    printf("Erro ao alocar a memória");
    exit(1);
}
```

```c
float *p;
p = (float *) calloc(50, sizeof(float));

if (!p)
{
    printf("Erro ao alocar a memória");
    exit(1);
}
```

Subsistema de Alocação Dinâmica

- malloc x calloc
  - Ambas alocam memória e retornam um ponteiro para o início da memória alocada
  - A função calloc ‘inicializa’ a memória após sua alocação
    - Coloca zero em todas as posições de memória alocada
  - Caso a memória alocada com malloc seja acessada logo após sua alocação, o retorno será lixo de memória
  - Por inicializar a memória, a função calloc é mais lenta que a malloc

Subsistema de Alocação Dinâmica

- A linguagem C oferece um subsistema para alocação dinâmica, cujas principais funções são:
  - calloc
  - malloc

  Aloca memória
  - **free**

  Libera memória
  - realloc

  Realoca a quantidade de memória alocada, para mais ou para menos

As funções pertencem a biblioteca stdlib.h

Subsistema de Alocação Dinâmica

- Função free
  - *void *free(void *ptr);*
  - Devolve ao heap a memória apontada por *ptr*, tornando a memória disponível para alocação futura.
  - A função *free* só deve ser usada com um ponteiro que foi previamente alocado com uma das funções do sistema de alocação dinâmica (malloc(), calloc() ou realloc()).

Subsistema de Alocação Dinâmica

- Função free
  - *void *free(void *ptr);*

```
float *p;
p = (float *) malloc(50*sizeof(float));

if (p == NULL)
{
     printf("Erro ao alocar a memória");
     exit(1);
}

free(p);
```

  - Libera na heap o espaço apontado por *p* (50*sizeof(float) bytes)

Subsistema de Alocação Dinâmica

- Função realloc
  - *void *realloc(void *ptr, size_t size);*
  - Modifica o tamanho da memória previamente alocada apontada por *ptr* para aquele especificado por *size*.
    - *size* pode ser maior ou menor que o original
    - Se *size* for zero, a memória é liberada
  - A função retorna um ponteiro para o bloco redimensionado de memória
    - Pode ser necessário copiar dados para outro bloco
    - Pode não haver memória necessária para a realocação e, neste caso, a função retorna nulo.

Subsistema de Alocação Dinâmica

- Função realloc
  - *void *realloc(void *ptr, size_t size);*

```c
float *p;
p = (float *) malloc(50*sizeof(float));

if (p == NULL)
{
    printf("Erro ao alocar a memória");
    exit(1);
}

p = realloc(p, 1000*sizeof(float));
if (p == NULL)
{
    printf("Erro ao realocar a memória");
    exit(1);
}
free(p);
```

Alocação Dinâmica

- Exercício:
  - Faça uma função que recebe dois vetores de inteiros, com qualquer número de elementos cada. A função deve imprimir todos os valores presentes nos dois vetores. Ex: se v1={19, 5, 2, 6} e v2={5, 0, 9, 4, 18, 56} deverá ser impresso somente o valor 5. O número de elementos de cada vetor deverá ser fornecido pelo usuário. Considere que os valores não se repitam dentro do vetor.
    - *Para essa atividade não precisa modularizar. Fazer no arquivo main.c*

Alocação Dinâmica

- Exercício:
  - Implementar um programa que cria um vetor dinâmico com o número de elementos informado pelo usuário.
    a) Inicializar os valores randomicamente.
       - v[i] = rand()%100;
    b) O usuário poderá alterar o tamanho do vetor quantas vezes desejar. Caso o novo tamanho seja menor que o anterior, as posições deverão ser inicializadas randomicamente.
    c) Liberar memória
    d) Após cada alocação/inicialização, chamar a função imprime_vetor.

    - *Para essa atividade não precisa modularizar. Fazer no arquivo main.c*

Alocação Dinâmica

- Exercício:
  - Implementar um programa que calcule a multiplicação de matrizes. O usuário deverá informar as dimensões da matriz A (MA) e da matriz B (MB) e receber o resultado do produto MA x MB (caso seja possível).
    - *Para essa atividade não precisa modularizar. Fazer no arquivo main.c*