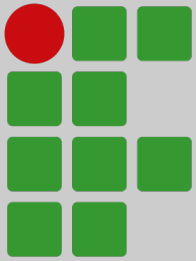


INSTITUTO FEDERAL

Campus Charqueadas

Alocação Dinâmica de Memória

Prof. Silvana Teodoro

silvanateodoro@charqueadas.ifsul.edu.br

Alocação dinâmica

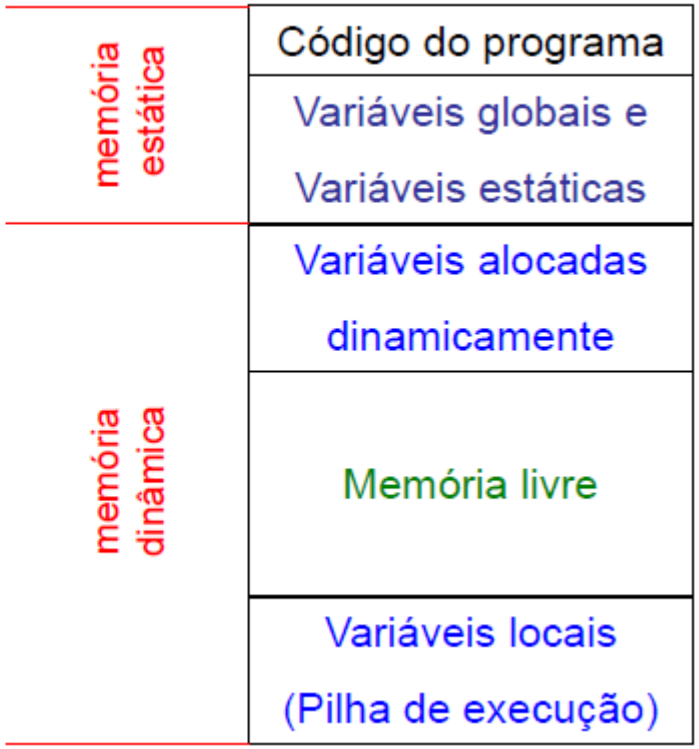
- Uso da memória:
  - uso de variáveis globais (e estáticas):
    - espaço reservado para uma variável global existe enquanto o programa estiver sendo executado
  - uso de variáveis locais:
    - espaço existe apenas enquanto a função que declarou a variável está sendo executada
    - liberado para outros usos quando a execução da função termina
  - variáveis globais ou locais podem ser simples ou vetores:
    - para vetor, é necessário informar o número máximo de elementos pois o compilador precisa calcular o espaço a ser reservado

Alocação dinâmica

- Uso da memória:
  - alocação dinâmica:
    - espaço de memória é requisitado em tempo de execução
    - espaço permanece reservado até que seja explicitamente liberado
      - depois de liberado, espaço estará disponibilizado para outros usos e não pode mais ser acessado
      - espaço alocado e não liberado explicitamente, será automaticamente liberado ao final da execução

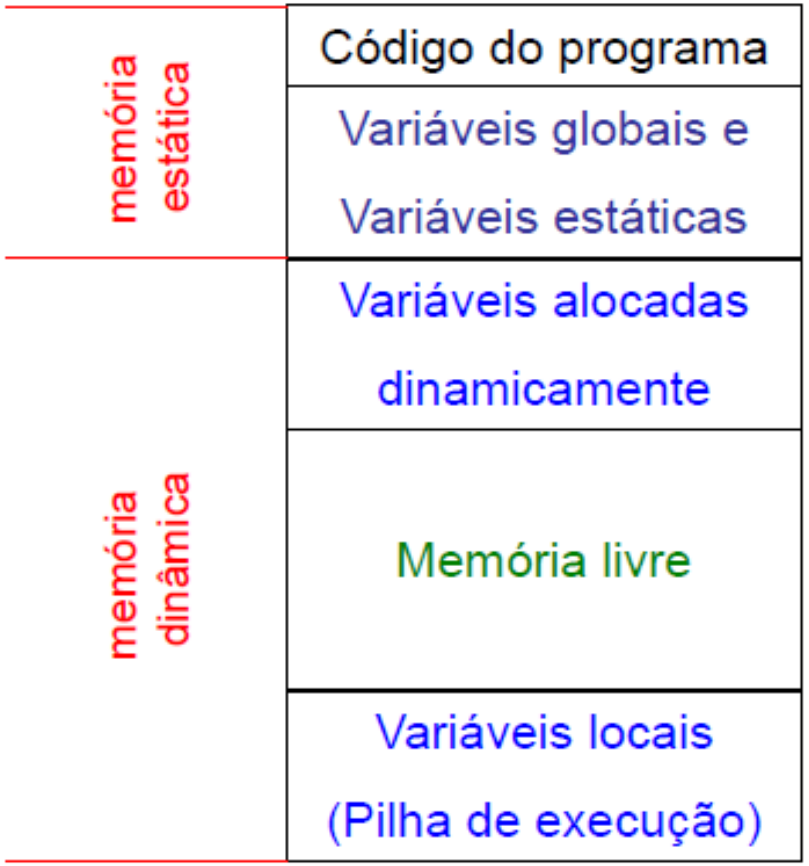
Alocação dinâmica

- Uso da memória:
  - memória estática:
    - código do programa
    - variáveis globais
    - variáveis estáticas
  - memória dinâmica:
    - variáveis alocadas dinamicamente
    - memória livre
    - variáveis locais



Alocação dinâmica

- Uso da memória:
  - alocação dinâmica de memória:
    - usa a memória livre
    - se o espaço de memória livre for menor que o espaço requisitado, a alocação não é feita e o programa pode prever tratamento de erro
  - pilha de execução:
    - utilizada para alocar memória quando ocorre chamada de função:
      - sistema reserva o espaço para as variáveis locais da função
      - quando a função termina, espaço é liberado (desempilhado)
    - se a pilha tentar crescer mais do que o espaço disponível existente, programa é abortado com erro



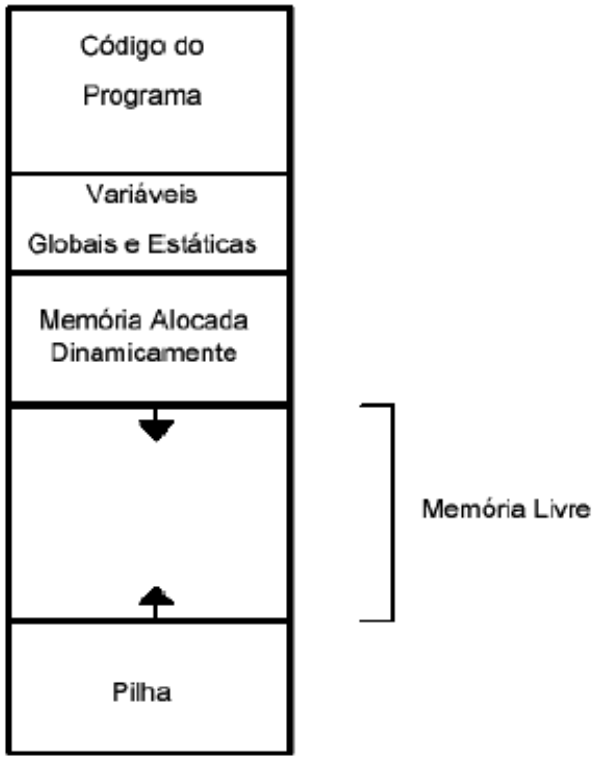
Alocação dinâmica

- Funções da biblioteca padrão “stdlib.h”
  - contém uma série de funções pré-definidas:
    - funções para tratar alocação dinâmica de memória
    - constantes pré-definidas
    - ....

Alocação dinâmica

```
void * malloc(int num_bytes);

void free(void * p);
```



Alocação dinâmica

- Função “**malloc**”:
  - recebe como parâmetro o número de bytes que se deseja alocar
  - retorna um ponteiro genérico para o endereço inicial da área de memória alocada, se houver espaço livre:
    - ponteiro genérico é representado por `void*`
    - ponteiro é convertido automaticamente para o tipo apropriado
    - ponteiro pode ser convertido explicitamente
  - retorna um endereço nulo, se não houver espaço livre:
    - representado pelo símbolo `NULL`

Alocação dinâmica

- Função “**sizeof**”:
  - retorna o número de bytes ocupado por um tipo
- Função “free”:
  - recebe como parâmetro o ponteiro da memória a ser liberada
    - a função free deve receber um endereço de memória que tenha sido alocado dinamicamente

Alocação dinâmica

- **Teste de tamanho:** execute o código abaixo...

```c
#include<stdio.h>

typedef struct{
   int dia, mes, ano;

}data;



main(){
  printf("sizeof(data) = %d\n", sizeof(data));
}
```

Alocação dinâmica

- Exemplo:
  - alocação dinâmica de um vetor de inteiros com 10 elementos
    - malloc retorna o endereço da área alocada para armazenar valores inteiros
    - ponteiro de inteiro recebe endereço inicial do espaço alocado

```
int *v;
v = (int *) malloc(10*sizeof(int));
```

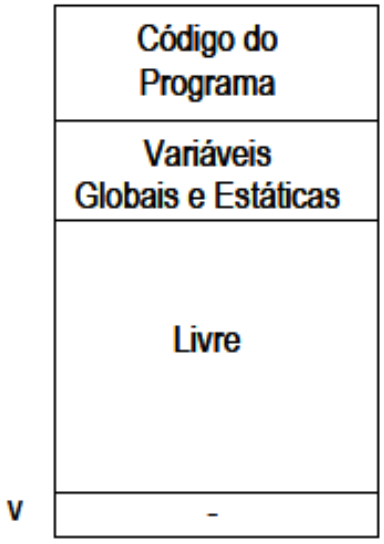
Alocação dinâmica

- Exemplo (cont.):

```
v = (int *) malloc(10*sizeof(int));
```

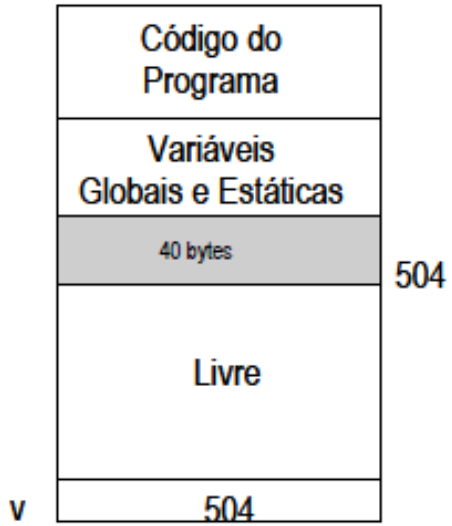
1 - Declaração: int *v

Abre-se espaço na pilha para o ponteiro (variável local)



2 - Comando: v = (int *) malloc (10*sizeof(int))

Reserva espaço de memória da área livre e atribui endereço à variável



Alocação dinâmica

- Exemplo (cont.):
  - v armazena endereço inicial de uma área contínua de memória suficiente para armazenar 10 valores inteiros
  - v pode ser tratado como um vetor declarado estaticamente
    - v aponta para o inicio da área alocada
    - v[0] acessa o espaço para o primeiro elemento
    - v[1] acessa o segundo
    - .... até v[9]

Alocação dinâmica

- Exemplo (cont.):
  - tratamento de erro após chamada a malloc
    - imprime mensagem de erro
    - aborta o programa (com a função exit)

```
...
v = (int*) malloc(10*sizeof(int));
if (v==NULL)
{
   printf("Memoria insuficiente.\n");
   exit(1);   /* aborta o programa e retorna 1 para o sist. operacional */
}
...

free(v);
```

Alocação dinâmica

```c
#include <stdlib.h>

int main ( void )
{
   float *v;
   float med, var;
   int i,n;

   printf("Entre n e depois os valores\n");
   scanf("%d",&n);
   v = (float *) malloc(n*sizeof(float));
   if (v==NULL) { printf("Falta memoria\n"); exit(1); }

   for ( i = 0; i < n; i++ )
      scanf("%f", &v[i]);

   med = media(n,v);
   var = variancia(n,v,med);

   printf ( "Media = %f   Variancia = %f  \n", med, var);
   free(v);
   return 0;
}
```

Atividade

- Escrever um programa que leia e ordene um vetor de estruturas contendo N valores reais e N nomes (N máximo 1000 elementos). Os nomes não devem ultrapassar 50 caracteres.

- O vetor resultante deve ser ordenado pelos valores numéricos, de forma crescente, a partir da primeira posição do vetor.

- Para ordenar o vetor compare cada elemento com o seguinte e troque-os de posição se necessário. Faça isto da primeira a última posição do vetor. Se, nesta varredura, houver ao menos uma troca de posições, será necessário repetir o procedimento. Quando realizar uma troca, não esqueça de trocar números e nomes.

Alocação dinâmica de matriz

```
float **v;  /* ponteiro para a matriz */
int   i;

/* aloca as linhas da matriz */
v = (float **) malloc (1000, sizeof(float *));
if (v == NULL) {
   printf ("** Erro: Memoria Insuficiente **");
   return (NULL);
 }
/* aloca as colunas da matriz */
for ( i = 0; i < 1000; i++ ) {
   v[i] = (float*) malloc (10, sizeof(float));
   if (v[i] == NULL) {
      printf ("** Erro: Memoria Insuficiente **");
      return (NULL);
     }
}
```

```
for (i=0; i<1000; i++) free (v[i]);  /* libera as linhas da matriz */
free (v);     /* libera a matriz */
```

Referências

- MIZRAHI, V. V. **Treinamento em linguagem C.** São Paulo: Makron Books, 1990.
- Material didático do Departamento de Informática da PUC-Rio (2014).
- Material didático do prof. Dr. Daniel Caetano (2012).