Computação 2

Aula 9

Alocação Dinâmica

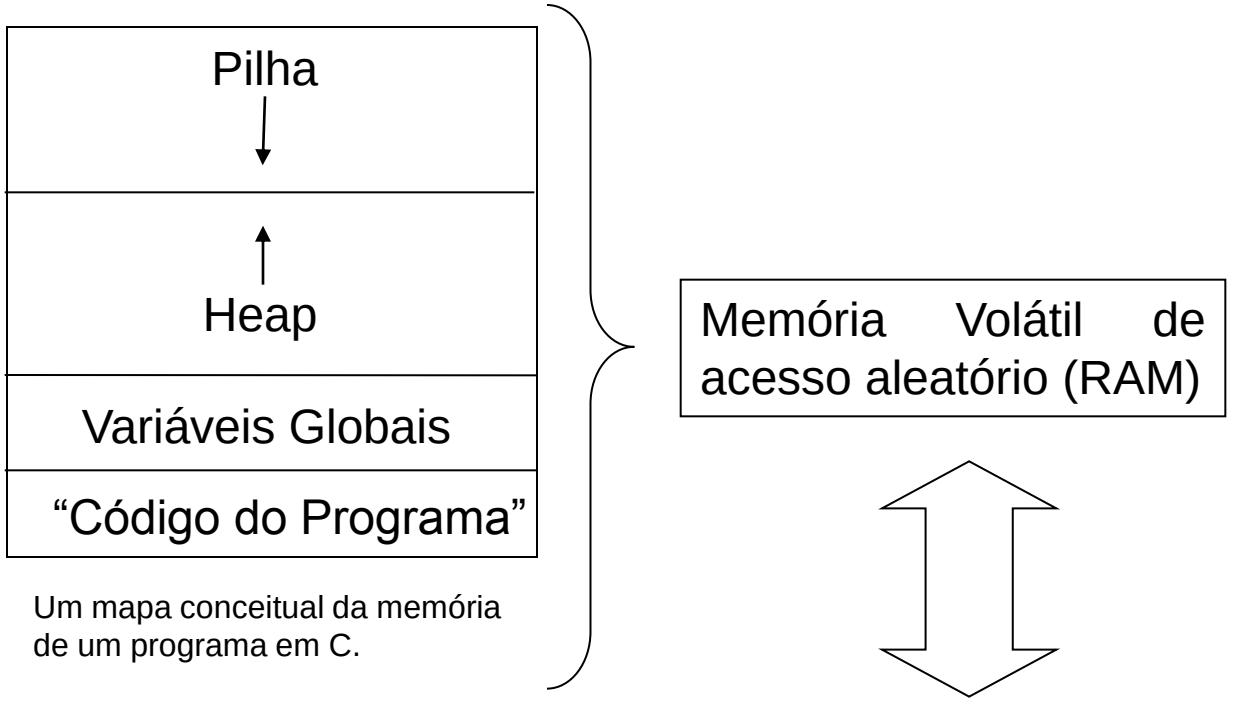
Profª. Fabiany
fabianyl@utfpr.edu.br

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Alocação Dinâmica

- Os ponteiros fornecem o suporte necessário para o sistema de alocação dinâmica de C.
- Alocação dinâmica é o meio pelo qual o programa pode obter memória enquanto está em execução.
- A memória alocada pelas funções de alocação dinâmica de C é obtida do *heap* – a região de memória livre que está entre seu programa e a área de armazenamento permanente e a pilha.

Memória



Um mapa conceitual da memória de um programa em C.

Alocação Dinâmica

- As principais funções para alocação dinâmica de C consiste nas funções **malloc()** e **free().**
- A função malloc() aloca memória e a função free() a libera.
- A função **malloc()** tem esse protótipo:

```
void *malloc(size_t número de bytes);
```

O número de bytes é número de bytes de memória que você quer alocar.

Alocação Dinâmica

- A função **malloc()** devolve um ponteiro do tipo void, assim é possível atribuí-la a qualquer tipo de ponteiro.

- Após uma chamada bem sucedida, **malloc()** devolve um ponteiro para o primeiro byte da região de memória alocada do heap. Se não há memória disponivel para alocar, ocorre uma falha de alocação e **malloc()** devolve nulo.

Alocação Dinâmica

- **Exemplo:**

```
char *p;
 p = malloc(1000);     //aloca 1000bytes de memória
```

```
int p;
p = malloc(50 * sizeof(int));   //aloca espaço para 50 inteiros
```

Exemplo 1

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

 int main()
 {
   // Definição de um ponteiro para a caracter
   char *s;

   // Variável inteira e de registro
   register int t;

   // Alocação de memóriana Heap para comportar 80 bytes (i.e. 80 caracteres)
   s = malloc ( 80 * sizeof ( char ) );

   if ( s )
   {
       printf ( "Ok Memoria alocada \n" );
   }

   // Se a memória não foi alocada
   if ( !s )
   {
     // Mensagem de falha
     printf ( "Falha na solicitação de memória. \n" );

     // Sair do programa
     exit( 1 );
  }
```

```
  // Mensagem e leitura de string!!!
  printf ( "Digite uma frase: \n" );

  // Se lê o ponteiro de caracteres tal qual se lê um vetor de caracteres
  // mesmo porque ponteiros para vetores de caracteres podem ser vistos como tal
  gets ( s );

  // Repetir do último elemento do "vetor" até o primeiro elemento.
  // Obs: strlen fornece o tamanho de "vetores“

  for ( t = strlen (s) -1;  t >= 0; t-- )
  {
     // Imprime um caracter na tela.
      putchar ( s [ t ] );
   }

   // Pula linha
   printf ( " \n " );

    // Libera a memória alocada para s.
    free ( s );

    system ("Pause");

    return 0;
}
```

Exemplo 2

```
#include <stdio.h>
#include <stdlib.h>

struct Pessoa
{
  int Idade;
  char Nome [100];
};

int main ()
{
   struct Pessoa *ListaP;
   ListaP = NULL;

   int quantidade = 0, conta  = 0,  i = 0;
   float media      = 0;

   printf ( "Quantas pessoas você irá cadastrar? \n" );
   scanf ( "%d", &quantidade );

   ListaP = malloc ( quantidade * sizeof ( struct Pessoa ) );

   printf ( " \n " );
   for ( i = 0; i < quantidade; i++ )
   {
      printf ( " Digite o nome da %d a. pessoa: \n ", i+1 );
      scanf ( "%s", ListaP[i].Nome );                    // gets ...

      printf ( "Digite a idade da %d a. pessoa: \n", i+1 );
      scanf ( "%d", &ListaP[i].Idade );

      conta = conta + ListaP[i].Idade;
      printf("\n");
   }
   media = (float)(conta / quantidade);

   printf ( " A média de idade é: %.2f anos. \n", media );
   printf ( "\n" );

   free ( ListaP );

   system ( "Pause" );
   return 0;

}
```

Referências Bibliográficas

- Baseado nos slides do Professor Jean Simão disponível em: http://www.pessoal.utfpr.edu.br/jeansimao/Fundamentos1/Fundamentos1.htm

- **SHILDT**, H. C, Completo e Total, 3a edição, rev. e atual. Ed. Makron. São Paulo, c1997.