Fundamentos de Programação 1

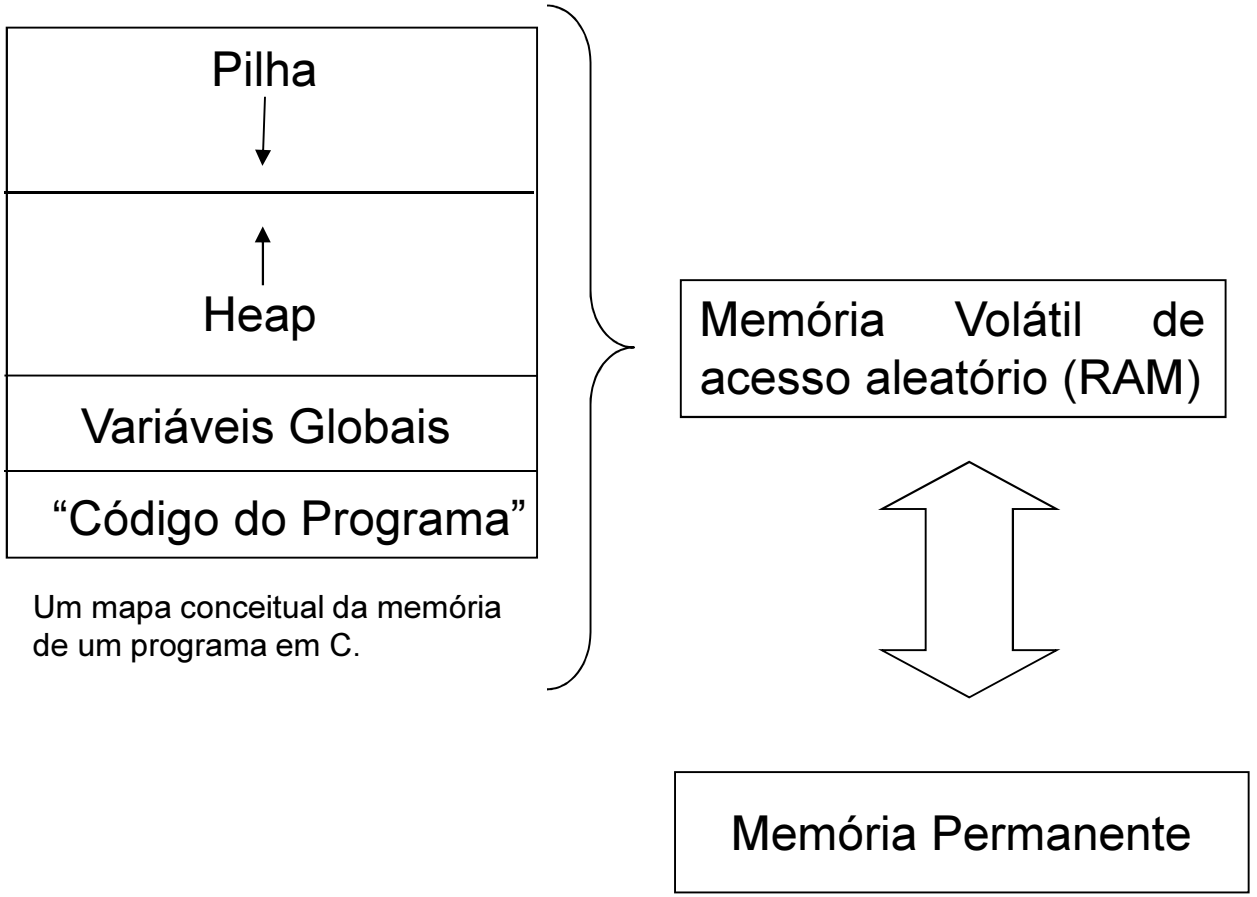
Linguagem C
“Alocação de Memória”.

Slides 16

Prof. SIMÃO

Jean Marcelo SIMÃO

Memória



Um mapa conceitual da memória de um programa em C.

Variáveis em C

Tipo de dados	Variação	Total de Bytes Utilizados
char	0 a 255	1
int	-32.768 a 32.767	2
short int	-128 a 127	1
unsigned int	0 a 65.535	2
long int	-4.294.967.296 a 4.294.967.296	4
float	Aproximadamente 6 dígitos de precisão	4
double	Aproximadamente 10 dígitos de precisão	8
long double	Aproximadamente 10 dígitos de precisão	10
void	-	0

Primeiro Exemplo

Alocação para Caracteres

Alocação de Memória

```
#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;
}
```

Segundo Exemplo

Alocação de Inteiros

```c
// Programa que imprime as potencias de 1 a 4 dos números de 1 a 10.
#include <stdio.h>
#include <stdlib.h>

// Cabeçalho de funções
int pwr ( int a, int b );           // função para o cálculo de potências
void table (int p[4][10]);          //  função para o armazenamento das potências em um tabela
void show (int p[4][10]);           //  função para mostrar o coteúdo da tabela.

int main()
{
   int *p = NULL;              // Ponteiro para inteiro

   // Alocação de memória equivalente ao espaço de 40 inteiros.
   // Sizeof(int) fornece o tamanho de um inteiro (normalmente dois bytes).
   p = malloc ( 40 * sizeof ( int ) );

   if ( !p )      // Se a memória não foi alocada...
   {
      printf ( "Falha na solicitação de memória. \n" );
      system ( "Pause" );
      exit ( 1 );
   }

   // Note que para ambas as funções abaixo se passa um 'vetor por parâmetro.
   // Na verdade é um ponteiro de inteiro que aponta para (o início de) um
   // conjunto de elementos que se comporta como um vetor...
   // Entretanto, as funções estão preparadas para receber matrizes 4x10...
   // mas o C consegue 'transformar' o ponteiro numa matriz...

   table ( p );                        // Cria a tabela de potências...
   show ( p );                         // Mostra a tabela de potências...

   free ( p );
   system ( "Pause" );
   return 0;
}
```

```
void show ( int p[4][10] )
{

    register int i, j;
    printf ( "%10s %10s %10s %10s \n", " N ", " N^2 ", " N^3 ", " N^4 " );

    for ( j = 1; j < 11;  j++ )
    {
       for ( i = 1; i < 5; i++ )
       {
          printf ( " %10d ", p[i-1][j-1] );
       }
       printf ( " \n " );
    }
    printf ( " \n " );

}
```

```
int pwr ( int a, int b )
{

    register int t = 1;

    for (; b; b--)
    {
       t = t * a;
    }

    return t;
}
```

```
void table ( int p [4][10] )
{
    register int i, j;

    for ( j = 1; j < 11; j++ )
    {
       for ( i = 1; i < 5; i++ )
       {
          p [i-1] [j-1] = pwr ( j , i );
       }
    }
}
```

Terceiro Exemplo

Alocação de Estruturas

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

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

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

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

}
```

Tipo Definido

typedef

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

struct Pessoa
{

  int Idade;
  char Nome [100];

};

int main()
{
   typedef int INTEIRO;

   typedef float REAL;

   typedef struct Pessoa CIDADAO;

   CIDADAO  *ListaC = NULL;

   INTEIRO  quantidade = 0;

   INTEIRO  conta      = 0;

   REAL     media    = 0;

   INTEIRO  i = 0;

   printf ( "Quantos cidadãos vc irá cadastrar? \n" );

   scanf ( "%d", &quantidade );

   ListaC = malloc ( quantidade * sizeof ( CIDADAO ) );

   printf ( "\n" );

   for ( i = 0; i < quantidade; i = i + 1 )
   {

      printf ( " Digite o nome do %d o. cidadão: \n ", i+1 );
      scanf ( "%s", ListaC[i].Nome );

      printf ( " Digite a idade do %d o. cidadão: \n ", i+1 );
      scanf ( "%d", &ListaC[i].Idade );

      conta = conta + ListaC[i].Idade;

      printf ( "\n" );
   }

   media = (REAL)(conta / quantidade);

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

   free ( ListaC );

   system ( "Pause" );

   return 0;

}
```