



Estruturas de Informação

Aula 5: Ponteiros & Alocação dinâmica

27/03/2008

Ponteiros (visão geral)

- Permite o armazenamento e manipulação de endereços de memória
- *Forma geral de declaração*
- *tipo *nome ou tipo* nome*
- Operadores de ponteiros (* e &)
- Aritmética de ponteiros (atribuição, adição e subtração)
- Relação entre ponteiros, vetores e matrizes
- Ponteiros para ponteiros ou indireção múltipla

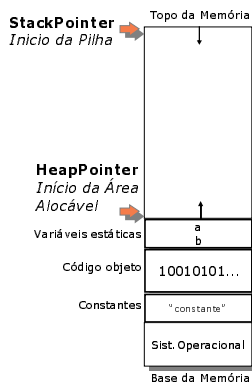
Alocação Dinâmica

- **Motivação**
 - Alocação fixa de memória (em tempo de desenvolvimento do programa) pode ser ineficiente
 - Por exemplo, alocar tamanhos fixos para nomes de pessoas pode inutilizar memória visto que existem tamanhos variados de nomes
 - Com alocação fixa em memória podemos ter espaços alocados na memória que não são utilizados
- **Solução: Alocação Dinâmica**
 - é um meio pelo qual o programa pode obter memória enquanto está em execução.
 - Obs.: tempo de desenvolvimento versus tempo de execução

Alocação da Memória

- Constantes: codificadas dentro do código objeto em tempo de compilação
- Variáveis globais (estáticas): alocadas no início da execução do programa
- Variáveis locais (funções ou métodos): alocadas através da requisição do espaço da pilha (stack)
- Variáveis dinâmicas: alocadas através de requisição do espaço do *heap*.
 - O heap é a região da memória entre o programa (permanente) e a stack
 - Tamanho do heap é a princípio desconhecido do programa

Memória



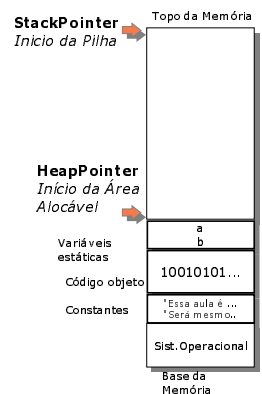
• Programa:

```
#include <stdio.h>
char *a, *b;

int func_A ()
{
    int local1, local2;
    - - -
}

void func_B ()
{
    int localA, localB;
    localA = func_A();
    localB = func_A();
}

main ()
{
    a = "Essa aula é legal";
    b = "Será mesmo?";
    func_B();
}
```



• Programa:

```
#include <stdio.h>
char *a, *b;

int func_A ()
{
    int local1, local2;
    ---
}
void func_B ()
{
    int localA, localB;
    localA = func_A();
    localB = func_A();
}
main ()
{
    a = "Essa aula é legal";
    b = "Será mesmo?";
    func_B();
}
```

StackPointer
Início da Pilha

Topo da Memória

HeapPointer
Início da Área Alocável

Variáveis estáticas
10010101...

Código objeto
'Essa aula é...'
'Será mesmo?'

Constantes

Sist. Operacional

Base da Memória

• Programa:

```
#include <stdio.h>
char *a, *b;

int func_A ()
{
    int local1, local2;
    ---
}
void func_B ()
{
    int localA, localB;
    localA = func_A();
    localB = func_A();
}
main ()
{
    a = "Essa aula é legal";
    b = "Será mesmo?";
    func_B();
}
```

StackPointer
Topo da Pilha

Topo da Memória

HeapPointer
Topo da Área Alocável

Variáveis estáticas
10010101...

Código objeto
'Essa aula é...'
'Será mesmo?'

Constantes

Sist. Operacional

Base da Memória

■ Programa:

```
#include <stdio.h>
char *a, *b;

int func_A ()
{
    int local1, local2;
    ---
}
void func_B ()
{
    int localA, localB;
    localA = func_A();
    localB = func_A();
}
main ()
{
    a = "Essa aula é legal";
    b = "Será mesmo?";
    func_B();
}
```

StackPointer
Topo da Pilha

Topo da Memória

HeapPointer
Topo da Área Alocável

Variáveis estáticas
10010101...

Código objeto
'Essa aula é...'
'Será mesmo?'

Constantes

Sist. Operacional

Base da Memória

■ Programa:

```
#include <stdio.h>
char *a, *b;

int func_A ()
{
    int local1, local2;
    ---
}
void func_B ()
{
    int localA, localB;
    localA = func_A();
    localB = func_A();
}
main ()
{
    a = "Essa aula é legal";
    b = "Será mesmo?";
    func_B();
}
```

StackPointer
Topo da Pilha

Topo da Memória

HeapPointer
Topo da Área Alocável

Variáveis estáticas
10010101...

Código objeto
'Essa aula é...'
'Será mesmo?'

Constantes

Sist. Operacional

Base da Memória

■ Programa:

```
#include <stdio.h>
char *a, *b;

int func_A ()
{
    int local1, local2;
    ---
}
void func_B ()
{
    int localA, localB;
    localA = func_A();
    localB = func_A();
}
main ()
{
    a = "Essa aula é legal";
    b = "Será mesmo?";
    func_B();
}
```

StackPointer
Topo da Pilha

Topo da Memória

HeapPointer
Topo da Área Alocável

Variáveis estáticas
10010101...

Código objeto
'Essa aula é...'
'Será mesmo?'

Constantes

Sist. Operacional

Base da Memória

■ Programa:

```
#include <stdio.h>
char *a, *b;

int func_A ()
{
    int local1, local2;
    ---
}
void func_B ()
{
    int localA, localB;
    localA = func_A();
    localB = func_A();
}
main ()
{
    a = "Essa aula é legal";
    b = "Será mesmo?";
    func_B();
}
```

StackPointer
Topo da Pilha

Topo da Memória

HeapPointer
Topo da Área Alocável

Variáveis estáticas
10010101...

Código objeto
'Essa aula é...'
'Será mesmo?'

Constantes

Sist. Operacional

Base da Memória

■ Programa:

```
#include <stdio.h>
char *a, *b;

int func_A ()
{
    int local1, local2;
    ---
}

void func_B ()
{
    int localA, localB;
    localA = func_A();
    localB = func_A();
}

main ()
{
    a = "Essa aula é legal";
    b = "Será mesmo?"
    func_B();
}
```

■ Programa:

```
#include <stdio.h>
char *a, *b;

int func_A ()
{
    int local1, local2;
    ---
}

void func_B ()
{
    int localA, localB;
    localA = func_A();
    localB = func_A();
}

main ()
{
    a = "Essa aula é legal";
    b = "Será mesmo?"
    func_B();
}
```

■ Programa:

```
#include <stdio.h>
char *a, *b;

int func_A ()
{
    int local1, local2;
    ---
}

void func_B ()
{
    int localA, localB;
    localA = func_A();
    localB = func_A();
}

main ()
{
    a = "Essa aula é legal";
    b = "Será mesmo?"
    func_B();
}
```

■ Programa:

```
#include <stdio.h>
char *a, *b;

int func_A ()
{
    int local1, local2;
    ---
}

void func_B ()
{
    int localA, localB;
    localA = func_A();
    localB = func_A();
}

main ()
{
    a = "Essa aula é legal";
    b = "Será mesmo?"
    func_B();
}
```

■ Programa:

```
#include <stdio.h>
char *a, *b;

int func_A ()
{
    int local1, local2;
    ---
}

void func_B ()
{
    int localA, localB;
    localA = func_A();
    localB = func_A();
}

main ()
{
    a = "Essa aula é legal";
    b = "Será mesmo?"
    func_B();
}
```

Alocação Dinâmica

- void *malloc (tamanho numero_bytes)
 - Retorna um ponteiro genérico para a área alocada
 - Retorna NULL se não for possível alocar
 - Usar type casting para especificar um tipo
 - V = (int *) malloc (sizeof (int));

Exemplo:

```

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

char *p;
int *q;

main ()
{
  p = (char *) malloc(1000);
  // Aloca 1000
  // bytes de RAM

  q = (int *) malloc(50*sizeof(int));
  // Aloca espaço
  // para 50 inteiros.
}

```

The diagram illustrates the memory layout. At the top is the **Topo da Memória** (Top of Memory). Below it is the **StackPointer** (Topo da Pilha). The **HeapPointer** (Topo da Área Alocável) is shown below the stack. The memory is divided into sections: **Variáveis estáticas** (Static variables), **Código objeto** (Object code), and **Sist. Operacional** (Operating system). The **Base da Memória** (Base of Memory) is at the bottom. The diagram also shows a block of **50*int = 200 bytes** and a block of **1000 bytes** in the heap area, with pointers **p** and **q** pointing to their respective locations.

Alocação Dinâmica (2)

- `void free (void *p)`
 - Devolve a memória previamente alocada para `p`
 - O ponteiro `p` deve ter sido alocado dinamicamente

Ambiente de Desenvolvimento Eclipse

- Downloads: www.eclipse.org (Eclipse IDE for C/C++ Developers)
- Disponível para linux e windows
- Para instalação do plugin C/C++ Development Kit (caso já tenha o eclipse para Java):
 - Menu Help -> Software Updates -> Find and Install -> Search for new features to install -> Europa Discovery Site
 - Escolha um mirror site (por exemplo Brazil Universo Online)
 - Escolha o plugin (C/C++ development)
- Se usar Windows:
 - Necessário CygWin ou Mingw
 - CygWin (gcc, gdb, etc), categoria "Devel"