Linguagem C: Ponteiros - Alocação Dinâmica

Prof. Paulo R. S. L. Coelho
paulo@facom.ufu.br

Faculdade de Computação
Universidade Federal de Uberlândia

GEQ007

Organização

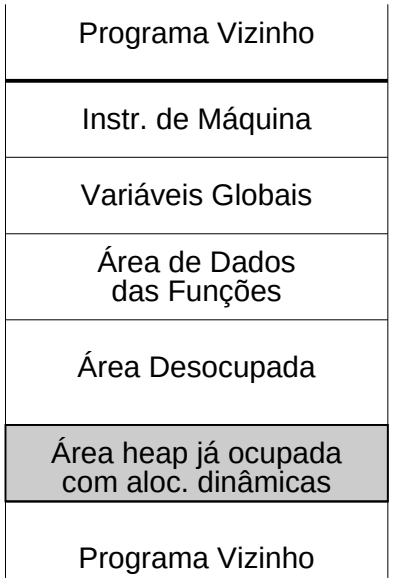
Organização

Alocação Dinâmica de Memória I

- Utilizado quando se deseja alocar espaço para variável em tempo de execução.
- Utilizado principalmente para variáveis indexadas (vetores e matrizes) e estruturas.
- A variável deve ser do tipo ponteiro.
- A reserva de memória pode ser feita pelo uso da função `malloc()`, pertencente à biblioteca `stdlib.h`.
- Essa função recebe como parâmetro o número de bytes a ser reservado.
- A região de memória ocupada pelo programa destinada a alocações dinâmicas é denominada *heap*.

Alocação Dinâmica de Memória II

- A reserva é feita nessa região, ocupando o número de bytes passado pela função, em **endereços contíguos**.
- A função retorna o endereço do 1^o byte reservado.
- A variável ponteiro alvo da alocação deve receber esse valor retornado.



Layout de um programa na memória

Organização

Alocação Dinâmica de Vetores I

- Considere o seguinte exemplo:

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

int main() {
  int m, A[300], B[300], C[300];

  printf("Entre tamanho dos vetores: ");
  scanf("%d", &m);

  printf("\nVetor A: ");
  for (i = 0; i < m; i++)
    scanf("%d", &A[i]);

  printf("\nVetor B: ");
  for (i = 0; i < m; i++)
    scanf("%d", &B[i]);

  printf("\nVetor C: ");
  for (i = 0; i < m; i++)
    C[i] = A[i] > B[i] ? A[i] : B[i];
```

Alocação Dinâmica de Vetores II

```
  for (i = 0; i < m; i++)
    printf("%d ", C[i]);

  return 0;
}
```

- Observe que foram utilizadas 900 posições de memória antes mesmo que o valor de m seja conhecido. Se m = 2, por exemplo, estamos ”desperdiçando” 894 posições.
- A alocação dinâmica permite criar o vetor somente após conhecermos seu real tamanho:

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

int main() {
  int m, *A, *B, *C;

  printf("Entre tamanho dos vetores: ");
  scanf("%d", &m);

  A = (int *) malloc(m * sizeof(int));
  B = (int *) malloc(m * sizeof(int));
```

Alocação Dinâmica de Vetores III

```
  C = (int *) malloc(m * sizeof(int));

  printf("\nVetor A: ");
  for (i = 0; i < m; i++)
    scanf("%d", &A[i]);

  printf("\nVetor B: ");
  for (i = 0; i < m; i++)
    scanf("%d", &B[i]);

  printf("\nVetor C: ");
  for (i = 0; i < m; i++)
    C[i] = A[i] > B[i] ? A[i] : B[i];

  for (i = 0; i < m; i++)
    printf("%d ", C[i]);

  free(A);
  free(B);
  free(C);
  return 0;
}
```

Alocação Dinâmica de Vetores IV

- Na atribuição:
  `A = (int *) malloc (m * sizeof(int));`
  A função `malloc()` reserva um espaço contíguo de **m*sizeof(int)** bytes e retorna o endereço inicial desse espaço.
- Assim, a partir desse momento, a variável A passa a atuar como uma variável indexada (vetor) comum.
- O fator de conversão `(int *)` deve ser usado, uma vez que o ponteiro retornado por essa função não tem tipo.
- Por fim, a função `free()` libera a memória reservada e apontada pelo ponteiro passado como argumento.

Organização

Alocação Dinâmica de Estruturas I

- Estruturas também podem ser alocadas dinamicamente.
- Por exemplo, considere a seguinte estrutura:

```
typedef struct st st;
struct st {
  int a;
  float b;
};
...
st *p;
```

- O comando `p = (st *) malloc(sizeof(st));` aloca espaço para uma estrutura **st**, cujo endereço é atribuído à variável **p**.

Alocação Dinâmica de Estruturas II

- As seguintes atribuições podem ser feitas aos campos dessa estrutura:
  `(*p).a = 2; (*p).b = 3.4;`

- A linguagem C estabelece outra forma de se referenciar os campos de uma estrutura apontada por um ponteiro.

- Dessa forma, as atribuições anteriores podem ser expressas da seguinte maneira:
  `p->a = 2; p->b = 3.4;`
  ou seja, `(*p).` pode ser substituído por `p->`

Organização

Exercícios I

1. Faça um programa que leia o tamanho de um vetor de inteiros e reserve dinamicamente memória para esse vetor. Em seguida, leia os elementos desse vetor, imprima o vetor lido e mostre o resultado da soma dos números ímpares presentes no vetor.
2. Faça um programa que crie uma estrutura `hidrocarboneto` com os campos `C` e `H` como inteiros correspondentes à quantidade de carbonos e hidrogênios, respectivamente.
   A função principal deve alocar dinamicamente uma variável do tipo `hidrocarboneto`. Em seguida, devem ser lidas as quantidades de carbonos e hidrogênios e impressa a massa atômica do composto.

Exercícios II

3. Considere a seguinte estrutura:

```
typedef struct aluno aluno;
struct aluno {
  char nome[30];
  float media;
  int faltas;
};
```

Faça um programa que leia informações de `n` alunos em um vetor alocado dinamicamente. Em seguida, imprima as informações lidas na ordem decrescente das médias dos alunos.