

Programação de Computadores II

Cap. 5 – Alocação Dinâmica

Livro: Waldemar Celes, Renato Cerqueira, José Lucas Rangel. *Introdução a Estruturas de Dados*, Editora Campus (2004)
Slides adaptados dos originais dos profs.: Marco Antonio Casanova e Marcelo Gattass (PUC-Rio)

Referências

Waldemar Celes, Renato Cerqueira, José Lucas Rangel, *Introdução a Estruturas de Dados*, Editora Campus (2004)

Capítulo 5 – Vetores e alocação dinâmica

Tópicos

- Alocação dinâmica
- Vetores locais e funções

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

memória estática	Código do programa
	Variáveis globais e Variáveis estáticas
memória dinâmica	Variáveis alocadas dinamicamente
	Memória livre
	Variáveis locais (Pilha de execução)

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

memória estática	Código do programa
	Variáveis globais e Variáveis estáticas
	Variáveis alocadas dinamicamente
memória dinâmica	Memória livre
	Variáveis locais (Pilha de execução)

7

Alocação Dinâmica

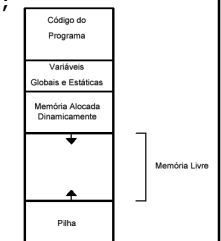
- Funções da biblioteca padrão “stdlib.h”
 - contém uma série de funções pré-definidas:
 - funções para tratar alocação dinâmica de memória
 - constantes pré-definidas
 -

8

Alocação Dinâmica

```
void * malloc(int num_bytes);
```

```
void free(void * p);
```



9

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 para o tipo apropriado, usando a conversão temporária (cast). Ex: float *v= (float *) malloc (1000 * sizeof(float));
 - retorna um endereço nulo, se não houver espaço livre:
 - representado pelo símbolo NULL
- 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

10

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));
```

11

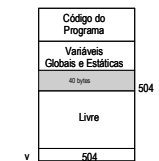
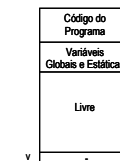
Alocação Dinâmica

- 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



12

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 início da área alocada
 - v[0] acessa o espaço para o primeiro elemento
 - v[1] acessa o segundo
 - até v[9]

13

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);
```

```
#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;
}
```

Vetores Locais a Funções

- Área de memória de uma variável local:
 - só existe enquanto a função que declara a variável estiver sendo executada
 - requer cuidado quando da utilização de vetores locais dentro de funções
- Exemplo:
 - produto vetorial de dois vetores **u** e **v** em 3D, representados pelas três componentes x, y, e z

$$\mathbf{u} \times \mathbf{v} = \{ u_y v_z - v_y u_z, \quad u_z v_x - v_z u_x, \quad u_x v_y - v_x u_y \}$$

16

Vetores Locais a Funções

```
float* prod_vetorial (float* u, float* v)
{
    float p[3];
    p[0] = u[1]*v[2] - v[1]*u[2];
    p[1] = u[2]*v[0] - v[2]*u[0];
    p[2] = u[0]*v[1] - v[0]*u[1];
    return p;
}
```

- variável p declarada localmente:
 - área de memória que a **variável p** ocupa deixa de ser válida quando a função `prod_vetorial` termina
 - função que chama `prod_vetorial` não pode acessar a área apontada pelo valor retornado

..'

Vetores Locais a Funções

```
float* prod_vetorial (float* u, float* v)
{
    float *p = (float*) malloc(3*sizeof(float));
    p[0] = u[1]*v[2] - v[1]*u[2];
    p[1] = u[2]*v[0] - v[2]*u[0];
    p[2] = u[0]*v[1] - v[0]*u[1];
    return p;
}
```

- variável p alocada dinamicamente
 - área de memória que a **variável p** ocupa permanece válida mesmo após o término da função `prod_vetorial`
 - função que chama `prod_vetorial` pode acessar o ponteiro retornado
 - problema - alocação dinâmica para cada chamada da função:
 - ineficiente do ponto de vista computacional
 - requer que a função que chama seja responsável pela liberação do espaço

Vetores Locais a Funções

```
void prod_vetorial (float* u, float* v, float* p)
{
    p[0] = u[1]*v[2] - v[1]*u[2];
    p[1] = u[2]*v[0] - v[2]*u[0];
    p[2] = u[0]*v[1] - v[0]*u[1];
}
```

- espaço de memória para o resultado passado pela função que chama:
 - função `prod_vetorial` recebe três vetores,
 - dois vetores com dados de entrada
 - um vetor para armazenar o resultado
 - solução mais adequada pois não envolve alocação dinâmica

19

Exercício

- Todo ano um concurso de programação premia os participantes que obtêm a maior média ponderada em uma bateria de dois testes
- Escreva uma função que recebe sete parâmetros:
 - o inteiro `n` indicando a quantidade de participantes do concurso
 - o ponteiro `inscr` para o vetor de inteiros que contém as inscrições desses participantes
 - o ponteiro `t1` para o vetor de reais que contém a nota de cada participante no primeiro teste
 - o inteiro `p1` que indica o peso dessa nota na média ponderada
 - o ponteiro `t2` para o vetor de reais que contém a nota de cada participante no segundo teste
 - o inteiro `p2` que indica o peso dessa nota na média ponderada
 - um ponteiro para a variável inteira `tam`

Exercício

- A função deve:
 - calcular a média ponderada de cada participante
 - criar um novo vetor de inteiros alocado dinamicamente com o tamanho exato para conter apenas as inscrições dos participantes que obtiveram a maior média (pode haver empate)
 - armazenar no novo vetor as inscrições correspondentes (em qualquer ordem)
 - armazenar o tamanho do novo vetor na variável `tam`
 - retornar o ponteiro para o novo vetor
- Considere que para uma mesma posição do vetor (mesmo índice), a inscrição e as notas se referem a um mesmo participante. O protótipo da função é:
`int *premiados(int n, int *inscr, float *t1, int p1, float *t2, int p2, int *tam);`

Resumo

Funções para gerência de memória:

- | | |
|---------------------|---|
| <code>sizeof</code> | retorna o número de bytes ocupado por um tipo |
| <code>malloc</code> | recebe o número de bytes que se deseja alocar
retorna um ponteiro para o endereço inicial, ou
retorna um endereço nulo (NULL) |
| <code>free</code> | recebe o ponteiro da memória a ser liberada |

22