Alocação Dinâmica

Professor: Guilherme Corredato Guerino

Disciplina: Algoritmos e Técnicas de Programação

Ciência da Computação
Universidade Estadual do Paraná

Introdução

- Considerando o **estudo da disciplina** realizado até o momento, tem-se a **definição do espaço alocado para uma variável** no momento de sua declaração.

- Isso significa que quando uma variável é **declarada**, tem-se que a **quantidade de memória** utilizada pela mesma é **conhecida**.

Introdução

- No caso de um vetor, é preciso saber inicialmente **quantos elementos podem ser armazenados no mesmo**.

- A **quantidade de elementos** que podem ser armazenados **influencia** na **quantidade de memória alocada** para o vetor.

Alocação Estática

- O **tipo de alocação** abordado na disciplina até agora é conhecido como **alocação estática de memória**.

- Neste contexto, não importa se a variável é **simples**, **vetor**, **matriz** ou **registro**, todas elas foram **declaradas** e tiveram o **espaço alocado** de **maneira estática**.

Observações

- A linguagem C permite alocar espaços de memória em **dois momentos distintos**:
  - Compilação
  - Execução

- Na **compilação** acontece a **alocação estática**. Na **execução** acontece a **alocação dinâmica**.

Alocação Dinâmica

- Por meio da **alocação dinâmica**, é possível **requisitar espaço** para **variáveis simples** e **compostas**, tais como **vetores**, **matrizes** e **registros**.

- Tal requisição acontece durante a **execução do programa**.

Alocação Dinâmica

- Existem funções na biblioteca **<stdlib.h>** relacionadas com a **alocação dinâmica**.

- Tais funções realizam a **alocação**, a **liberação** e a **realocação** de memória em **tempo de execução**.

Alocação Dinâmica de Memória

- A função básica para **alocar memória** é a **malloc**. Tal função recebe o **número de bytes que se deseja alocar** como **parâmetro**.

- Se a alocação for **bem sucedida**, o **endereço inicial da área de memória alocada é retornado**.

Alocação Dinâmica de Memória

- Se **não existir espaço livre suficiente para ser alocado**, a função malloc retorna um **endereço nulo**, que é representado pelo termo **NULL**.

- O termo **NULL** está definido em **<stdlib.h>**.

Exemplo

```
include<stdio.h>
include<stdlib.h>
int main(){
  // declaração de um ponteiro
  int *v;
  // exemplo de alocação dinâmica
  v = (int*) malloc (10*4);
  return 0;
}
```

Observações

- A função malloc retorna um **espaço de memória** do tipo **void**.
- Para definir um tipo para o espaço, realiza-se a **conversão de tipo**.
- Tal conversão acontece pela "**inserção**" do tipo do dado entre parênteses, **antes da chamada da função malloc**.

Observações

- Se a alocação for bem-sucedida, **v receberá um espaço de memória** suficiente para armazenar 40 bytes.

- Supondo que um valor do tipo inteiro utiliza **4 bytes**, esse espaço é **suficiente** para armazenar **10 valores inteiros**.

Observações

- Se a alocação **não for** bem sucedida, **NULL** é retornado.

- É possível **checar o retorno** da função malloc. Dependendo do valor de retorno, é possível saber o **resultado da alocação**.

Exemplo

```
...
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 */
}
...
```

Liberação Dinâmica de Memória

- Para **liberar um espaço de memória** alocado de **forma dinâmica**, utiliza-se a função **free**.

- Essa função recebe o **ponteiro** que referencia o **espaço de memória** a ser **liberado** como **parâmetro**.

Exemplo

```
include<stdio.h>
include<stdlib.h>
int main(){
  int *v;
  v = (int*) malloc (10*4);
  free(v);
  return 0;
}
```

Continuação

- A **alocação dinâmica** introduz **flexibilidade** na **alocação de memória**.

- É possível **alocar** e **liberar** memória durante a **execução do programa**, inclusive de vetores, matrizes e registros.

Vetor

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

int main(){
   int *vetor;
   vetor = (int*) malloc (10*sizeof(int));
   free(vetor);

   return 0;
}
```

Exemplo

- **No trecho de código apresentado, as seguintes funções merecem destaque:**
  - **malloc():** realiza a **alocação** de um espaço de memória.
  - A **quantidade de bytes solicitados** é passada como **parâmetro**, **entre parênteses**.

Exemplo

- **No trecho de código apresentado, as seguintes funções merecem destaque:**
  - **sizeof():** retorna o **tamanho**, ou a **quantidade de bytes**, que uma **variável** de determinado **tipo** considera.
  - O **tipo do dado** é passado como **parâmetro**, entre **parênteses**.

Exemplo

- **No trecho de código apresentado, as seguintes funções merecem destaque:**
  - **free():** realiza a **liberação** de um espaço de memória, **alocado dinamicamente**.
  - A referência desse espaço (**ponteiro**) é passada como **parâmetro**, entre **parênteses**.

Exemplo

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

int main(){
   int *vetor;
   vetor = (int*) malloc (10*sizeof(int));
   free(vetor);

   return 0;
}
```

Observações

- No trecho de código do exemplo apresentado, o valor **10** representa a **dimensão do vetor**, determinado em **tempo de execução**.

- O valor **10** poderia ter sido fornecido pelo usuário.

Observações

- Assim, um espaço suficiente para armazenar 10 valores inteiros foi alocado.

- Após a alocação dinâmica, o **acesso** aos elementos do vetor é realizado da **mesma maneira**, para **vetores estáticos** e/ou **dinâmicos**.

Observações

- Na alocação de um **vetor dinâmico**, é necessário **declarar uma variável do tipo ponteiro**.

- Tal variável recebe o **valor do endereço do primeiro elemento do vetor**, alocado dinamicamente.

Observações

- A **área de memória** utilizada pelo vetor **permanece válida** até que seja **liberada**.

- A liberação acontece **após a execução** da função **free()**.

Realocação Dinâmica

- Além da **alocação** e da **liberação**, a Linguagem C ainda oferece um mecanismo para **realocação de um vetor dinâmico**.

- Em **tempo de execução**, pode-se **verificar** que a **dimensão escolhida** para o vetor tornou-se **insuficiente** ou **excessivamente grande**.

Realocação Dinâmica

- O **redimensionamento** do vetor possibilita "**resolver a situação**".

- A função **realloc()** possibilita a **realocação** de um vetor, **preservando o conteúdo** do mesmo.

Realocação Dinâmica

- Os elementos do vetor **permanecem válidos** após a realocação.

- No trecho de código a seguir, **m** representa a nova dimensão do vetor.

```
// realocação do vetor
vetor = (int*) realloc (vetor, m*sizeof(int));
```