Alocação dinâmica de memória

Prof. Leandro O. Freitas
leandro@politecnico.ufsm.br

Alocação dinâmica

- Muitas vezes o programador não sabe de antemão a quantidade de dados que seu programa deverá manter em memória para correto processamento.
  - Programas que devem manter na memória dados de um arquivo de tamanho desconhecido para seu processamento.
    - Exemplo: editor de textos.
    - Qual o número máximo de caracteres que deve suportar?

Alocação dinâmica

Sistemas para Internet
UFSM

- Como declarar variáveis para conter esses dados?
  - Uma forma de solucionar esse problema é declarar variáveis com um determinado tamanho, razoavelmente grande, e, caso esse tamanho seja insuficiente, o programa deve informar o usuário sobre essa limitação e abortar a execução.

- Isto leva a dois grandes problemas:
  - Espaço alocado demasiado, ou seja, desperdício de memória;
  - Ou espaço insuficiente para alocação de todos os dados.

Alocação dinâmica

- Outra solução é o programa alocar memória do sistema dinamicamente (durante a sua execução), conforme a necessidade do usuário.

- Desta forma, o programa utilizará somente a memória necessária para o processamento que irá realizar e poderá processar um volume de dados limitado somente pela capacidade de memória do sistema.

Alocação dinâmica

Sistemas para Internet
UFSM

- Definição:
  - **Alocação Dinâmica** é o processo de solicitar e utilizar memória durante a execução de um programa. Ela é utilizada para que um programa em C utilize apenas a memória necessária pra sua execução, sem desperdícios.

- Sendo assim, a alocação dinâmica de memória deve ser utilizada quando não se sabe, por algum motivo ou aplicação, a quantidade de memória necessária para o armazenamento de algum(ns) valores.

Alocação dinâmica

Sistemas para Internet
UFSM

- As bibliotecas *stdio.h e stdlib.h* da linguagem C possui funções para solicitação dinâmica de memória ao sistema operacional. As principais funções são:
  - Para alocação: malloc()
  - Para liberação: free()

- Considere que existem situações em que o computador não possui memória disponível para ser alocada.
  - Neste caso, um pedido de alocação será recusado. Isto deve ser tratado pelo desenvolvedor.

Alocação dinâmica

- malloc (*<quantidade solicitada>*);
  - Pedido de alocação de memória.

- Sintaxe:

```
int *v;
v = malloc (4); //aloca um espaço de 4 bytes na memória;
v = malloc (10 * 4); //aloca um espaço de 40 bytes na memória;
v = malloc (10 * sizeof(int)) //aloca um espaço para 10 valores inteiros;
v = (int *) malloc (10 * sizeof(int)) ; // especifica o tipo de ponteiro de retorno;
```

- Para que seja possível determinar qual o tamanho em *bytes* desejado, é utilizado o comando ***sizeof***.

Alocação dinâmica

Sistemas para Internet
UFSM

- O retorno da função **malloc()** é um ponteiro **genérico que deve ser configurado** para o tipo alocado. Por exemplo:

```
int *p;
p = (int*) malloc (sizeof (int));
```

- Essa instrução solicita alocação de memória para o tamanho de um ***inteiro*** e recebe como retorno de malloc() um ***endereço*** disponível para um elemento do tipo inteiro.

- Assim, o valor do ponteiro ***p*** é o endereço da memória onde o espaço foi alocado. Nele, é possível armazenar valores inteiros.

- Caso o pedido de memória seja recusado, ***p*** receberá o valor **NULL** (nulo). Assim, deve-se testar se um pedido de memória foi aceito, verificando se a variável que recebe o retorno de malloc() é igual a **NULL**.

```
if (p == NULL) {
   printf("Memória insuficiente \n");
}
```

Alocação dinâmica

Sistemas para Internet
UFSM

- free()
  - Liberação da memória alocada dinamicamente.

- Sintaxe
  - free (endereço);

- Toda memória alocada **dinamicamente** deve ser liberada ANTES do término da execução de um programa, caso contrário, ela não estará disponível para outras aplicações ou programas.

Alocação dinâmica

Sistemas para Internet
UFSM

- Atenção: você é responsável por liberar a memória que você alocou!

- Uma vez que a única maneira de chegar a uma memória alocada dinamicamente é através de um ponteiro, a liberação desse endereço de memória deve ser realizado a partir dele. Exemplo:

```
free(p);
```

Alocação dinâmica

- O trecho de código a seguir solicita a alocação dinâmica de um vetor de inteiros com 25 posições, atribui um valor para a posição 10 e depois libera a área de memória.

```
int *vector = NULL; /* declaração do ponteiro */
vector = (int*) malloc(25 * sizeof(int)); /* alocação de memória para o vector */
vector[10] = 34; /* altera o valor da posição dez para trinta e quatro */
free(vector); /* liberta a área de memória alocada */
```

Alocação dinâmica: exemplo com vetor

Sistemas para Internet
UFSM

```c
int main(){
    int *vetor, i, n;//Declara um vetor sem o tamanho, através de ponteiro
    printf("Informe o tamanho do vetor: ");
    scanf("%d", &n); //Define tamanho
    vetor = (int*)malloc(n * sizeof(int)); //Aloca memória para o vetor
    if(vetor != NULL){  //verifica se há espaço livre na memória
        printf("Informe os elementos do vetor: \n");
        for(i=0; i<n; i++){                    //Se houver espaço de memória para
            scanf("%d", &vetor[i]); //ou (vetor + i)o vetor, é possível inserir
        }                                      elementos e manipulá-los conforme
                                                necessidade do usuário
        printf("Valores inseridos no vetor: \n");
        for(i=0; i<n; i++){
            printf("%d\n", vetor[i]); //ou *(vetor + i)
        }
    }
    else{ //Caso não exista espaço disponível na memória
        printf("Memoria insuficiente. \n\n");
    }

    free(vetor);//libera espaço de memória usada pelo vetor.
    return 0;
}
```

Exercício

Sistemas para Internet
UFSM

- Desenvolva um algoritmo para:
  - Criar dinamicamente um vetor de n elementos na função principal;
  - Passar o vetor por parâmetro para uma função para inserir elementos;
  - A função deve apresentar os elementos ao usuário;
  - Liberar memória antes do término do programa.