Alocação Dinâmica de Memória

Sérgio Portari Jr
Algoritmos e Estruturas de Dados
UEMG – Campus de Frutal

Alocação Estática x Dinâmica

- C: dois tipos de alocação de memória: **Estática e Dinâmica**

- Na alocação estática, o espaço para as variáveis é reservado no início da execução, não podendo ser alterado depois
  - int a; int b[20];

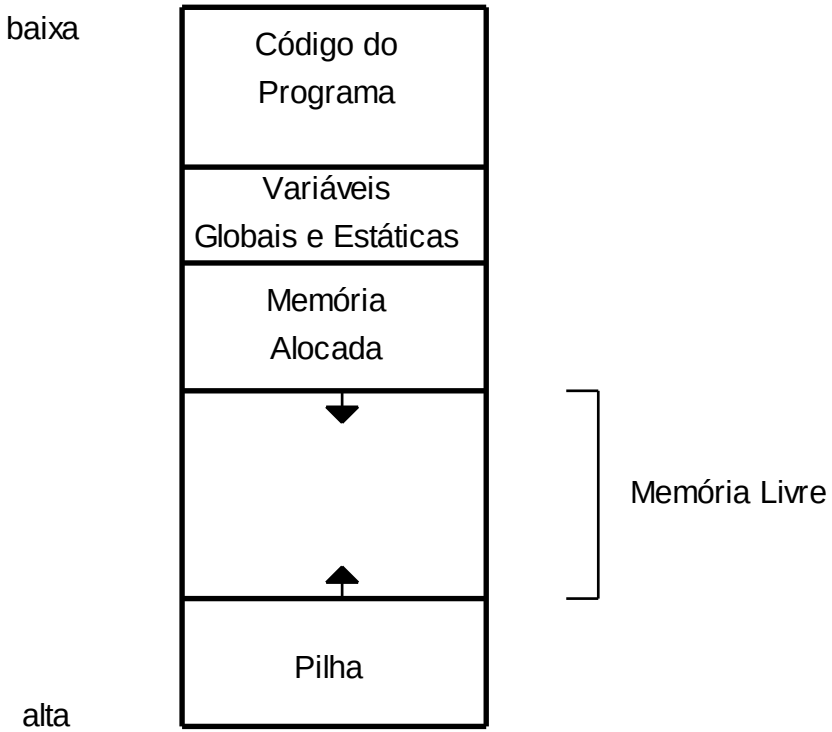
- Na alocação dinâmica, o espaço para as variáveis pode ser alocado dinamicamente durante a execução do programa

Alocação Dinâmica

- As variáveis alocadas dinamicamente são chamadas de **Apontadores** (*ponteiros*) pois na verdade elas armazenam o endereço de memória de uma variável

- A memória alocada dinamicamente faz parte de uma área de memória chamada ***heap***
  - Basicamente, o programa aloca e desaloca porções de memória do heap durante a execução

Esquema de Memória



Esquema da memória do sistema

Acesso a partir de Apontadores

- Acessar o valor da variável: endereço de memória armazenado
  - Exemplo:

```
int *p;
p = &a;
```

- Acessar o conteúdo que associado ao endereço de memória armazenado
  - Exemplo:

```
int *p;
p = &a;
*p=45;
```

Liberação de Memória

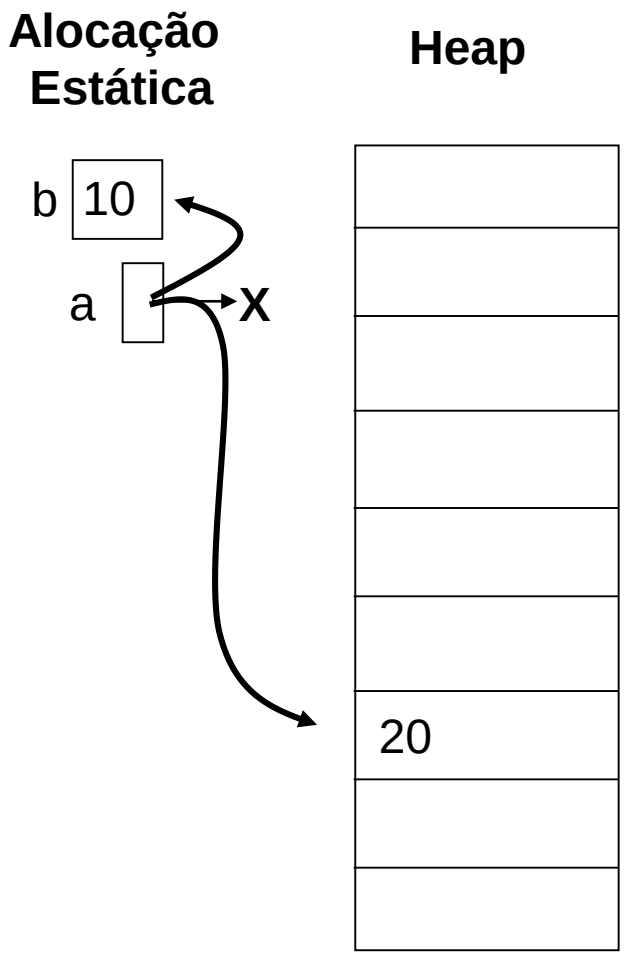
- A memória deve ser liberada após o término de seu uso
- A liberação deve ser feita por quem fez a alocação:
  - Estática: compilador
  - Dinâmica: programador

Apontadores – Notação (c)

- definição de p como um apontador para uma variável do tipo Tipo
  - Tipo *p;
- Alocação de memória para uma variável apontada por p
  - p = (Tipo*) malloc(sizeof(Tipo));
- Liberação de memória
  - free(p);
- Conteudo da variável apontada por P
  - *p;
- Valor nulo para um apontador
  - NULL;
- Endereço de uma variável a
  - &a;

Alocação Dinâmica

```
int *a, b;
...
b = 10;
a = (int *) malloc(sizeof(int));
*a = 20;
a = &b;
```



Erros Comuns

- **Esquecer de alocar memória e tentar acessar o conteúdo da variável**
- Copiar o valor do apontador ao invés do valor da variável apontada
- Esquecer de desalocar memória
  - Ela é desalocada ao fim do programa ou procedimento função onde a variável está declarada, mas pode ser um problema em loops
- Tentar acessar o conteúdo da variável depois de desalocá-la

Exemplo em C++

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
int *a,b;
b=10;
printf("\nantes da alocacao:");
printf("\nvalor de a=%d",*a);
printf("\nendereco de a=%p",a);
printf("\nvalor de b=%d",b);
printf("\nendereco de
   b=%p",&b);
a=(int *)malloc(sizeof(int));
printf("\ndepois da
   alocacao:");
printf("\nvalor de a=%d",*a);
printf("\nendereco de a=%p",a);
printf("\nvalor de b=%d",b);
printf("\nendereco de
   b=%p",&b);
*a=20;
printf("\ndepois de alocacao e
atribuicao:");
printf("\nvalor de a=%d",*a);
printf("\nendereco de a=%p",a);
printf("\nvalor de b=%d",b);
printf("\nendereco de b=%p",&b);
a=&b;
printf("\ndepois da alocacao e
apontar b:");
printf("\nvalor de a=%d",*a);
printf("\nendereco de a=%p",a);
printf("\nvalor de b=%d",b);
printf("\nendereco de b=%p",&b);
a=NULL;
free(a);
printf("\ndepois de liberar a
memoria:");
//printf("\nvalor de a=%d",*a);
//a nao existe mais, da erro
printf("\nendereco de a=%p",a);
    printf("\nvalor de b=%d",b);
    printf("\nendereco de
b=%p",&b);
    return 0;
}
```

Pergunta que não quer calar...

int *a não pode apontar um vetor de int?

- Em C, todo vetor é um ponteiro.
- Portanto pode-se fazer coisas como:

```
int a[10], *b;
b = a;
b[5] = 100;
printf("%d\n", a[5]);
printf("%d\n", b[5]);
```

100
100

```
int a[10], *b;
b = (int *) malloc(10*sizeof(int));
b[5] = 100;
printf("%d\n", a[5]);
printf("%d\n", b[5]);
```

42657
100

Obs. Não se pode fazer a = b no exemplo acima

Apontadores para Tipos Estruturados

- Apontadores são normalmente utilizados com tipos estruturados

```
typedef struct {
          int idade;
          double salario;
} funcionario;

funcionario *a,b;
a = (funcionario *) malloc(sizeof(funcionario));
a->idade = 30
a->salario = 800.87;
b.idade = 34;
b.salario = 1314.22;
printf("\nIdade *a: %d",a->idade); //acesso dinâmico
printf("\nSalario *a: %.2f",a->salario); //acess. dinâmico
printf("\nIdade b: %d",b.idade); //acesso estático
printf("\nSalario b: %.2f",b.salario); //acess. estático
```

Passagem de Parâmetros (C)

```
void SomaUm(int x, int *y)
{
      x = x + 1;
      *y = (*y) + 1;
      printf("Funcao SomaUm: %d %d\n", x, *y);
}

int main()
{
      int a=0, b=0;
      SomaUm(a, &b);
      printf("Programa principal: %d %d\n", a, b);
}
```

```
1  1
```

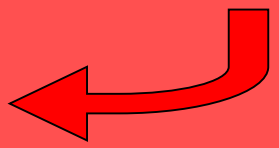
```
0  1
```

Passagem de Parâmetros

- E para alocar memória dentro de um procedimento para criar um vetor?
- Para realizar a alocação dinâmica em uma função precisamos passar um ponteiro de ponteiro.

```
void aloca(int *x, int n)
{
  x=(int *)malloc(n*sizeof(int));
  x[0] = 20;
}
int main()
{
  int *a;
  aloca(a, 10);
  a[1] = 40;
printf("a[0]=%d \n a[1]=%d",
a[0],a[1]);
}
```

Error!
Access Violation!



```
void aloca(int **x, int n)
{
  *x=(int *)malloc(n*sizeof(int));
  *x[0] = 20;
}
int main()
{
  int *a;
  aloca(&a, 10);
  a[1] = 40;
printf("a[0]=%d \n a[1]=%d",
a[0],a[1]);
}
```

OK

Exercício 1

1. Faça um programa que leia um valor *n,* crie dinamicamente um vetor de n elementos e passe esse vetor para uma função que vai ler os elementos desse vetor.

Exercício 2

- Criar um tipo que é uma estrutura que represente uma pessoa, contendo nome, data de nascimento e CPF.
- Criar uma variável que é um ponteiro para esta estrutura (no programa principal)
- Criar uma função que recebe este ponteiro e preenche os dados da estrutura
- Criar uma função que recebe este ponteiro e imprime os dados da estrutura
- Fazer a chamada a esta função na função principal