Alocação Dinâmica de Memória

Algoritmos e Estrutura de Dados II

DCC - UFMG

Alocação Estática x Dinâmica

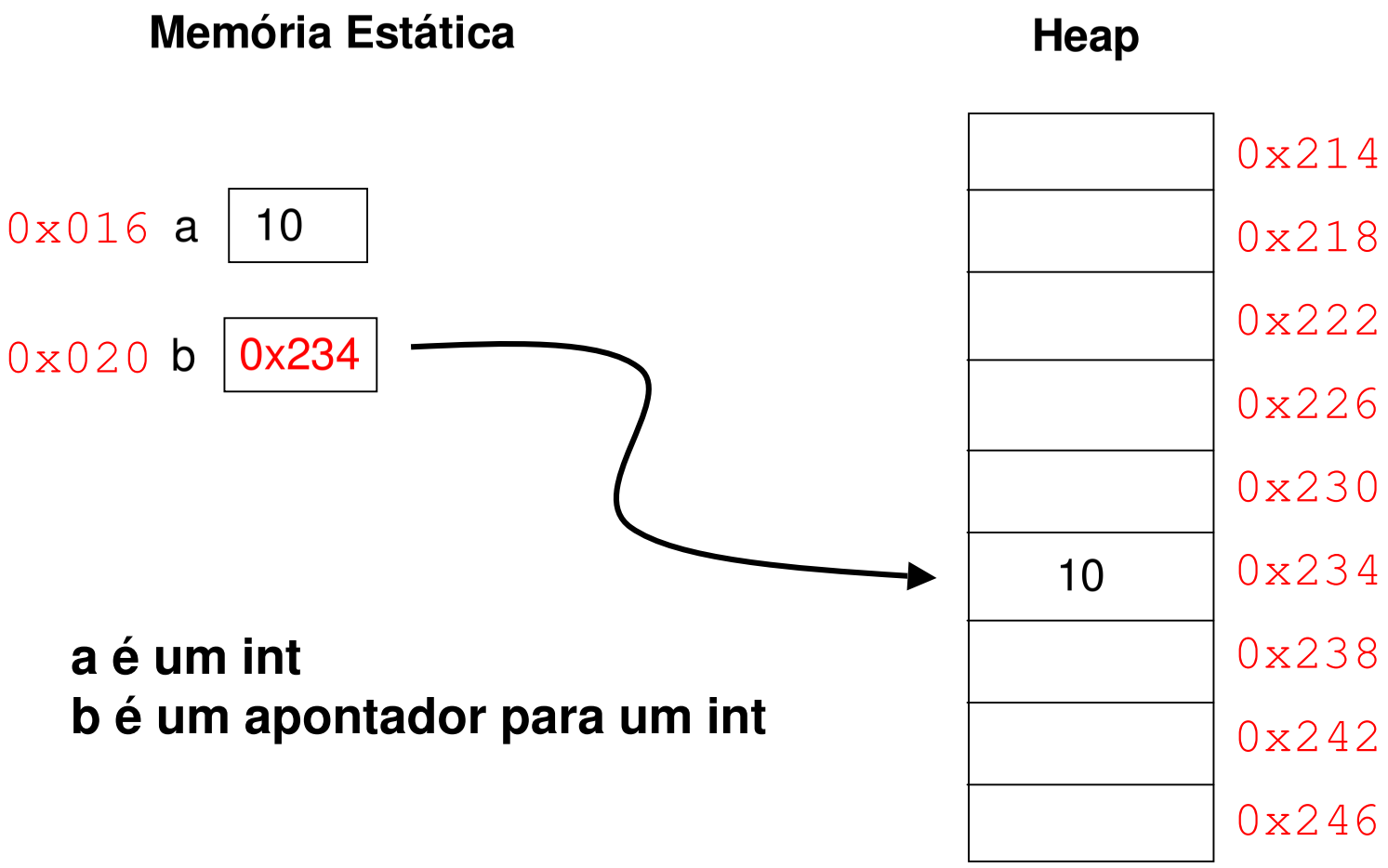
- Linguagens de programação como Pascal, C e C++ permitem dois tipos de alocação de memória: Estática e Dinâmica
- Na alocação estática, o espaço de memória para as variáveis é reservado no início da execução, não podendo ser alterado depois
  - int a; int b[20];
- Na alocação dinâmica, o espaço de memória para as variáveis pode ser alocado dinamicamente durante a execução do programa

Alocação Dinâmica

- A memória alocada dinamicamente é acessada através de **Apontadores** (*pointers*) que na verdade são variáveis que armazenam o endereço de uma área de memória

- 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

Alocação Dinâmica



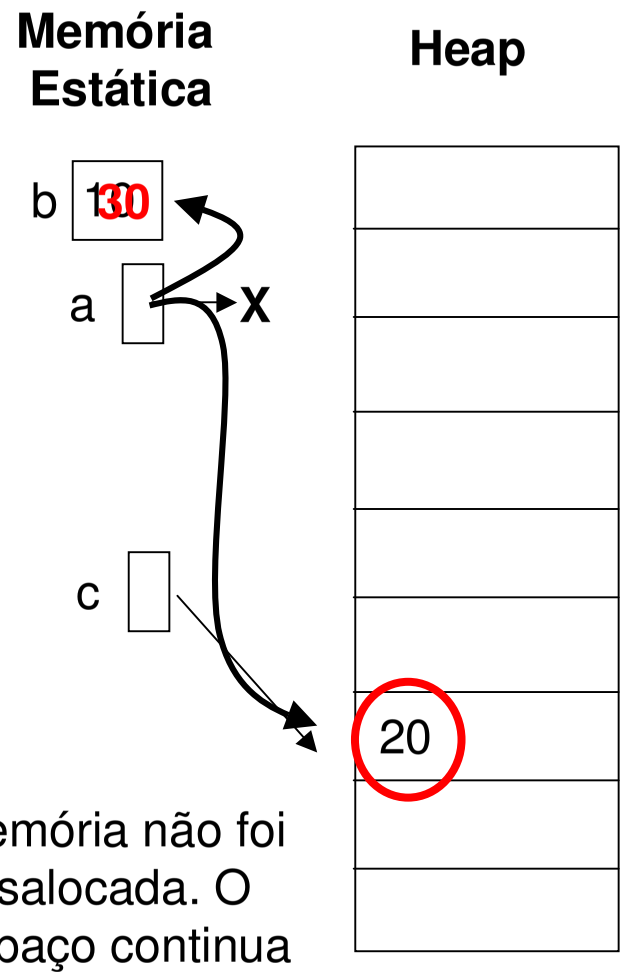
a é um int
b é um apontador para um int

Apontadores – Notação em C

- definição de p como um apontador para uma variável do tipo T
  - T *p;
- Alocação de memória para uma variável apontada por p
  - p = (T*) malloc(sizeof(T));
- Desalocação 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,*c, b;
...
b = 10;
a = (int *) malloc(sizeof(int));
c = a;
*a = 20;
free(a)
a = &b;
*a = 30;     // qual o valor de b?
```



Memória não foi desalocada. O espaço continua ocupado

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

Exercício: O que vai ser impresso?

```
double a;
double *p, *q;

a = 3.14;
printf("%f\n", a);
p = &a;
*p = 2.718;
printf("%f\n", a);
a = 5;
printf("%f\n", *p);
p = NULL;
p = (double *)malloc(sizeof(double));
*p = 20;
q = p;
printf("%f\n", *p);
printf("%f\n", a);
free(p);
printf("%f\n", *q);
```

Pergunta que não quer calar...

int *a não é a declaração de um vetor de int?

- Em C, todo vetor é um apontador.
- 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;
} TRegistro

TRegistro *a;
...
a = (TRegistro *) malloc(sizeof(TRegistro))
a->idade = 30;
a->salario = 80;
```

Passagem de Parâmetros

- Em pascal e C++, parâmetros para função podem ser passados por valor ou por referência
  - **Por valor:** o parâmetro formal (recebido no procedimento) **é uma cópia** do parâmetro real (passado na chamada)
  - **Por referência:** o parâmetro formal (recebido no procedimento) **é uma referência** para o parâmetro real (passado na chamada)
    - As modificações efetuadas acontecem no parâmetro real
- Em C só existe passagem por valor, logo deve-se implementar a passagem por referência utilizando-se apontadores

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

1 1

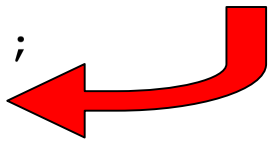
0 1

Passagem de Parâmetros

- E para alocar memória dentro de um procedimento?
  - Em pascal, basta passar a variável (apontador) como referência.
  - Em C também, mas como não há passagem por referência as coisas são um pouco mais complicadas

```
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;
}
```

Error!
Access Violation!



```
void aloca(int **x, int n)
{
  *x=(int *)malloc(n*sizeof(int));
  if (*x == NULL)
        printf("ERROR!!!!\n");
  else   *x[0] = 20;
}
int main()
{
  int *a, **b;
  b = &a;
  aloca(b, 10);
  a[1] = 40;
}
```

OK

Exercícios

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.

2. Declare um TipoRegistro, com campos *a* inteiro e *b* que é um apontador para char. No seu programa crie dinamicamente uma váriavel do TipoRegistro e atribua os valores 10 e ‘x’ aos seus campos.