Alocação dinâmica de vetores e matrizes

Prof. Thiago Felski Pereira, MSc.

Alocação dinâmica de vetores

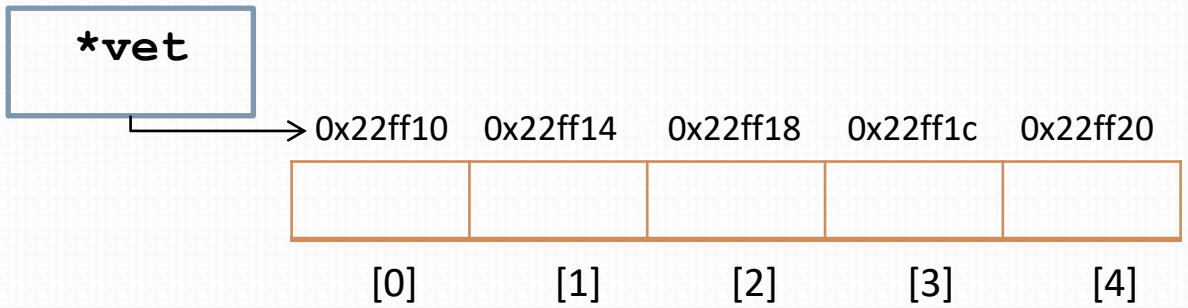
- Quando é necessário armazenar dados em memória sem que se conheça o tamanho ou a quantidade de dados necessários, pode-se recorrer ao recurso de alocação dinâmica de memória, que permite a alocação e liberação de áreas de memória a qualquer momento durante a execução do programa.

- O limite para alocação pode ser tão grande quanto a quantidade de memória física disponível no computador ou a quantidade de memória virtual.

Alocação dinâmica de vetores

- Um vetor dinamicamente alocado, ou vetor dinâmico, é um vetor cujo tamanho não é especificado ao se escrever o programa, mas é determinado durante a execução do programa.
- Uma variável vetor nada mais é que uma espécie de variável ponteiro que aponta para a primeira variável indexada do vetor.

```
int vet[5];
```



Alocação dinâmica de vetores

Tanto vet quanto ponteiro são variáveis do tipo ponteiro.

Aqui, a variável ponteiro passa a apontar para a posição para onde a variável vet está apontando.

Desta forma, pode-se utilizar a variável ponteiro para percorrer o vetor, pois ambas estão apontando para o mesmo endereço de memória.

```
int main() {
    int vet[10];
    int *ponteiro, i;
    ponteiro = vet;
    for (i=0;i<10;i++)
        ponteiro[i] = i * 2;
    for (i=0;i<10;i++)
        cout<<vet[i]<<"   ";
    return 0;
}
```

Restrição

- **Restrição encontrada**: por exemplo, se fosse criada uma variável do tipo ponteiro chamada `p2`, e ela estivesse apontando para algum valor, não se pode fazer a seguinte atribuição:

```
vet = p2;
```

- Isso acontece porque uma variável vetor não é do tipo `int*`, mas sim uma versão `const` de `int*`. O valor da variável `vet` não pode ser alterado.

Alocação dinâmica de vetores

- Até o momento, era necessário especificar o tamanho do vetor ao escrever o programa

- Quando não se sabia o tamanho de vetor necessário estimava-se o tamanho maior possível, o que poderia gerar dois problemas: a estimativa poderia ser baixa, ou o programa ter muitas posições não utilizadas, acarretando em um desperdício de memória

- Vetores dinâmicos evitam este problema, pois o usuário pode fornecer o tamanho do vetor ao iniciar a execução do programa

Alocação dinâmica de vetores

- Os operadores `new` e `delete` são essenciais para a criação dos vetores dinâmicos, pois é com eles que se solicita ao sistema operacional a alocação (`new`) ou liberação (`delete`) de memória. Estes operadores alocam ou liberam memória para comportar o tipo de dado especificado.

- É necessário tomar cuidado ao utilizar esse tipo de recurso, para não alocar memória de forma indiscriminada sem a devida liberação, pois se isso ocorrer, a memória ficará cheia de áreas reservadas mas sem uso o que representa “lixo” e não deve ocorrer.

Alocação dinâmica de vetores

Aqui foi declarada uma variável ponteiro do tipo float e alocado um espaço de memória do tamanho especificado pelo usuário. Também poderia ter sido feito: float vet = new float [TAM];

A variável ponteiro vet irá apontar para o primeiro endereço de memória alocado.

A variável ponteiro criada e o espaço alocado deverão ser do mesmo tipo.

Deve-se colocar os [] antes do nome do vetor pois eles dizem ao C++ que uma variável dinamicamente alocada foi eliminada, e o sistema então verifica o tamanho do vetor e remove aquela quantidade de variáveis indexadas.

```
int main() {
    int TAM, i;
    cout<<"Tamanho: ";
    cin>>TAM;
    float *vet;
    vet = new float[TAM];
    for (i=0;i<TAM;i++)
        cin>>vet[i];
    for (i=0;i<TAM;i++)
        cout<<vet[i]<<" ";
    delete []vet;
    return 0;
}
```

Exercício

- Verifique e entenda o que acontece ao final da execução do programa, a seguir:

```
#include <iostream>
using namespace std;

int* dobro (int a[], int tam){
    int *temp = new int[tam];
    int i;
    for (i=0;i<tam;i++)
        temp[i] = 2 * a[i];
    return temp;
}
```

```
int main () {
    int a[] = {1,2,3,4,5};
    int *b, i, tam = 5;

    b = dobro (a,tam);
    cout<<"Vetor A\n";
    for (i=0;i<5;i++)
        cout<<a[i]<<" ";

    cout<<"\nVetor B\n";
    for (i=0;i<tam;i++)
        cout<<b[i]<<" ";
    delete []b;
    return 0;
}
```

Alocação dinâmica de matrizes

- Também conhecidos como vetores dinâmicos multidimensionais
- Uma matriz dinâmica é um vetor de vetores
- Primeiro se cria um vetor dinâmico unidimensional de ponteiros, e após um vetor dinâmico para cada elemento do vetor
- Existem duas formas de alocar matrizes dinamicamente:

Matrizes: exemplo A

Declara um ponteiro para ponteiro e aloca uma área de tamanho LIN para dados do tipo ponteiro para inteiro

Aloca uma área de tamanho LIN*COL para dados do tipo inteiro e armazena seu endereço na primeira posição do vetor m

```
int main () {
    int LI = 3;
    int CO = 3;

    int **m = new int*[LI];
    m[0] = new int[LI*CO];

    for (int i=1; i<LI; i++){
        m[i] = m[i-1]+CO;
    }
```

Aloca para cada posição do vetor de ponteiros um vetor de inteiros, que corresponde as posições da matriz

Matrizes: exemplo A

Os laços demonstram que o uso da matriz, após alocada, é o mesmo

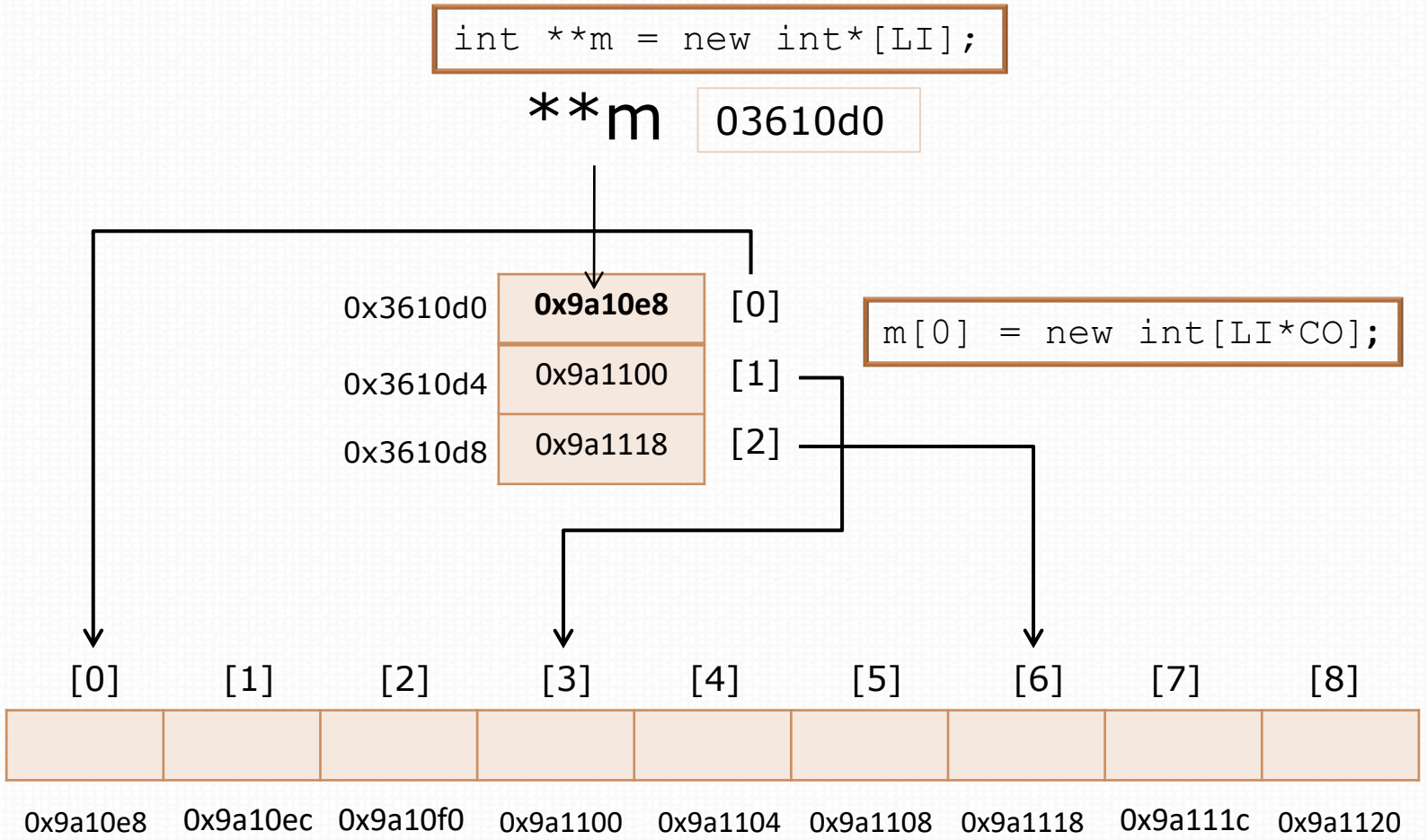
```
for (int i=0; i<LI; i++){
    for (int j=0; j<CO; j++){
        m[i][j] = i+j;
    }
}

for (int i=0; i<LI; i++){
    for (int j=0; j<CO; j++){
        cout<<m[i][j] <<" ";
    }
    cout<<"\n";
}

delete m[0];
delete m;
```

A primeira linha deleta a área alocada para a matriz, e na segunda a área alocada para o vetor de ponteiros

Matrizes: Exemplo A




```
for (int i=1; i<LI; i++)
        m[i] = m[i-1]+CO;
```

Matrizes: Exemplo B

Declara um ponteiro para ponteiro e aloca uma área de tamanho LIN para dados do tipo ponteiro para inteiro

Aloca para cada posição do vetor de ponteiros um vetor de inteiros, que corresponde as posições da matriz

```
int main () {

    int LI = 3;
    int CO = 3;

    int **m = new int*[LI];

    for (int i=0; i<LI; i++){
        m[i] = new int[CO];
    }

    //...laços de teste...

    for (int i=0; i<LI; i++){
        delete m[i];
    }
    delete m;

    return 0;
}
```

Libera cada um dos vetores de dados alocados

Libera o vetor de ponteiros

Matrizes: Exemplo B