Algoritmos e Estrutura de Dados - Alocação Dinâmica

Eduardo Augusto Costa Trindade

eduardo.trindade@ifmg.edu.br

Sistemas de Informação

Instituto Federal de Ciência e Tecnologia de Minas Gerais
Campus São João Evangelista

2021

Sumário

Introdução

- Sempre que escrevemos um programa é preciso reservar espaço para os dados que serão processados.
- Para isso usamos as variáveis.
- Toda variável deve ser declarada antes de ser usada.

Introdução

- Nem sempre é possível saber o quanto de memória um programa irá precisar.
- Uma solução simples para resolver esse problema poderia ser declarar um vetor bem grande.
- **Exemplo**: float alunos[1000];
- **Possíveis Problemas**: Desperdício de memória; tamanho insuficiente.

Introdução

Na declaração de um vetor:

- É feita uma reserva de quantidade de memória.
- A quantidade reservada é fixa.

Introdução

Pergunta: É possível fazer essa reserva de maneira variável? Se sim, como? Se não, por quê?

Introdução

Pergunta: É possível fazer essa reserva de maneira variável? Se sim, como? Se não, por quê?

Sim! Através da combinação do uso de **ponteiros** e **vetores**.

Relembrando

- **Variável**: é uma posição na memória previamente reservada e que pode ser usada para armazenar algum dado.
- **Ponteiro**: é uma variável que guarda um endereço de memória.
- **Vetor**: é um ponteiro que aponta para o primeiro elemento de um conjunto de dados que estão na memória.

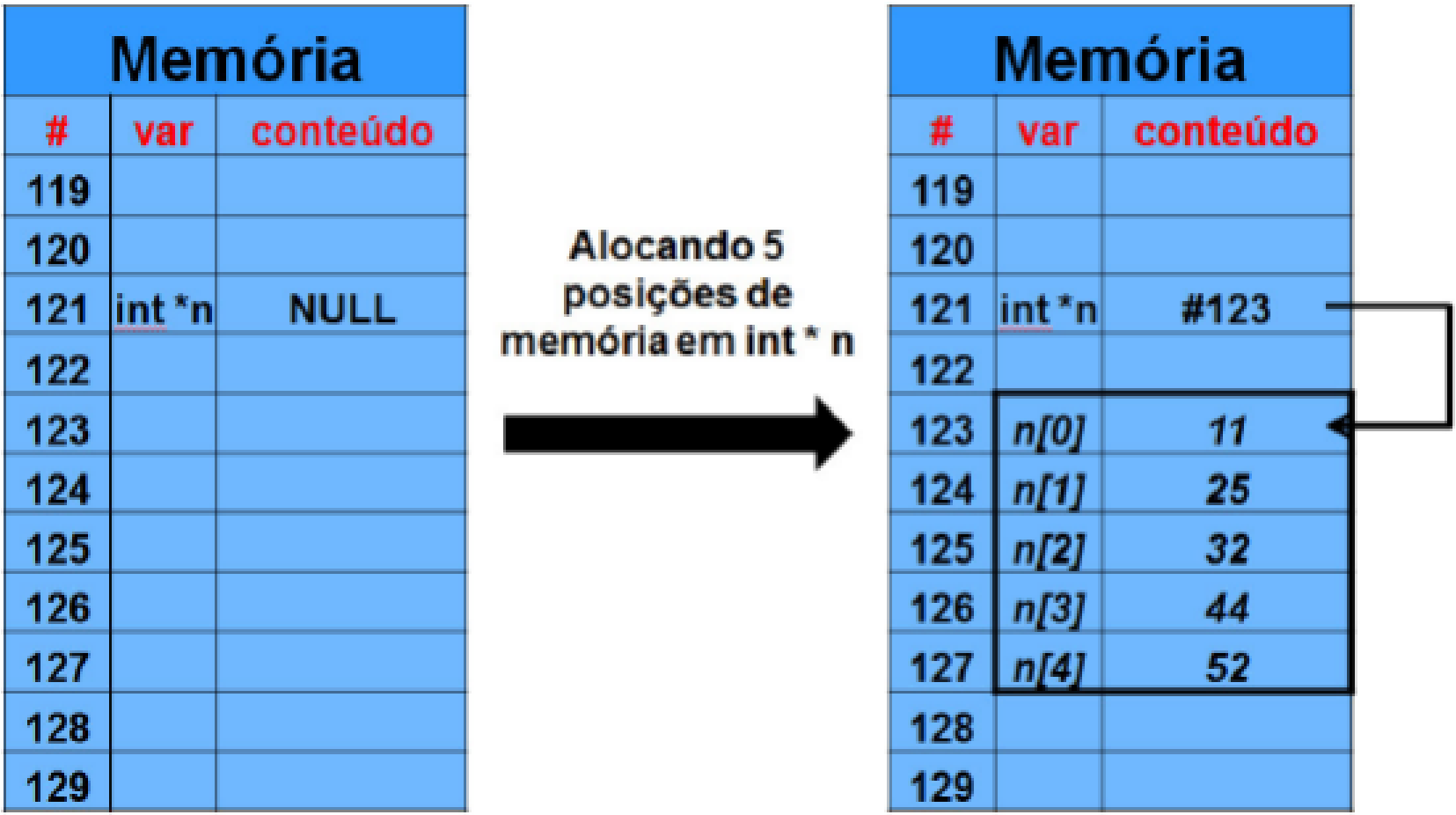
Alocação Dinâmica

- A linguagem C/C++ permite alocar dinamicamente blocos de memórias utilizando ponteiros.
- A esse processo dá-se o nome de **alocação dinâmica**.
- Esse tipo de alocação acontece quando o programa está sendo executado.

Alocação Dinâmica

- Ela é utilizada quando não se sabe ao certo quanto de memória será necessário para armazenar os dados.
- Deste modo evita-se o desperdício de memória.

Alocação Dinâmica



Alocação Dinâmica: Exemplos

- **Exemplo 1:** Se você deseja ler todas as linhas de um arquivo e armazená-los em um vetor de seu programa, o tamanho de memória necessário dependerá do tamanho do arquivo.

- **Exemplo 2:** Os processadores de texto, nos quais não sabemos a quantidade de caracteres que o utilizador vai escrever quando o programa estiver sendo executado.

Alocação Dinâmica: Stack

- É uma região da memória do seu computador que armazena variáveis locais criadas por cada função (incluindo a função *main()*).
- É uma estrutura de dados "LIFO"(última entrada, primeira saída).
- É gerenciada e otimizada pela própria CPU.
- Toda vez que uma função declara uma nova variável, ela é "colocada"na pilha.

Alocação Dinâmica: Stack

- Cada vez que uma função termina, todas as variáveis colocadas na pilha por essa função são liberadas.
- Neste caso, não é necessário alocar memória.
- Como é a CPU quem organiza a memória em pilha, a leitura e a escrita para pilha são muito rápidas.
- Quando uma função termina, todas as suas variáveis são retiradas da pilha.

Alocação Dinâmica: Stack

- Existe um limite para o tamanho das variáveis que podem ser armazenadas na pilha.
- Este não é o caso das variáveis alocadas no *heap*.

Alocação Dinâmica: *Heap*

- É uma região livre da memória, não gerenciada pela CPU.
- Uma vez que você alocou memória no *heap*, você é responsável por desalocar.
- O *heap* não possui restrições de tamanho no tamanho da variável.

Alocação Dinâmica: *Heap*

- A memória do *heap* é ligeiramente mais lenta para ser lida e gravada.
- Ao contrário da pilha, as variáveis criadas no *heap* são acessíveis por **qualquer função**.

Alocação Dinâmica: Operador **new**

- Realiza uma solicitação de alocação de memória no *heap*.

  Se houver memória suficiente disponível:

- o operador **new** alocará a memória.
- retornará o endereço de memória recém-alocada.
- um ponteiro deve ser usado para receber o retorno do operador **new**.

```
8   int *p1;
9   p1 = new int;
```

Alocação Dinâmica: Operador **new**

- O operador **new** produz um espaço de memória para uma nova variável.
- O tipo para esta nova variável deve ser especificado escrevendo o nome do tipo após o operador **new**.
- As variáveis que são criadas usando o operador **new** são chamadas **variáveis alocadas dinamicamente**, ou simplesmente **variáveis dinâmicas**.
- Essas variáveis são criadas e destruídas enquanto o programa está em **execução**.

Alocação Dinâmica: Exemplo

```cpp
#include <iostream>

using namespace std;

int main() {

    int *p1, *p2;
    p1 = new int;
    *p1 = 42;
    p2 = p1;
    cout << "*p1 = " << *p1 << endl;
    cout << "*p2 = " << *p2 << endl;
    *p2 = 53;
    cout << "*p1 = " << *p1 << endl;
    cout << "*p2 = " << *p2 << endl;
    p1 = new int;
    *p1 = 88;
    cout << "*p1 = " << *p1 << endl;
    cout << "*p2 = " << *p2 << endl;
  return 0;
}
```

Vetores Dinâmicos

- Um problema dos vetores é que você deve especificar o seu tamanho no programa.
- Mas você pode não saber qual o tamanho do vetor que você precisa até que o programa seja executado.
- **Exemplo:** Um vetor pode conter uma lista de números de identificação de alunos, mas o tamanho da classe pode ser diferente sempre que o programa é executado.

Vetores Dinâmicos

- Com os tipos de vetores vistos até agora, você deve estimar o maior tamanho possível.
- Suponha que o seu programa usa um vetor alocado dinamicamente para números de identificação do aluno.
- O número de alunos pode ser inserido como entrada para o programa. O vetor alocado dinamicamente pode ser criado para ter um tamanho exatamente igual ao número de alunos.
- Vetores alocados dinamicamente são criados usando o operador **new**.

Vetores Dinâmicos

- Como um vetor é um ponteiro, você pode usar o operador **new** para criar variáveis dinamicamente alocadas que são vetores, tratando esses vetores alocados dinamicamente como se fossem vetores comuns.
- Exemplo:

```
6
7     int tamanho;
8     cin >> tamanho;
9     int *vetor;
10    vetor = new int[tamanho];
11
```

- Para obter um vetor alocado dinamicamente de qualquer outro tipo, basta substituir o **int** pelo tipo desejado.

Vetores Dinâmicos

Observe que:

- O tipo de ponteiro para um vetor alocado dinamicamente é o mesmo que o tipo do ponteiro que você usaria para um único elemento do vetor.
- O ponteiro para o vetor é um ponteiro para a primeira variável indexada do vetor.
- O tamanho do vetor alocado dinamicamente é dado entre colchetes após o tipo. Se omitido, alocaria o armazenamento suficiente para apenas uma variável.

Vetores Dinâmicos

- A declaração de exclusão de um vetor alocado dinamicamente é feita da seguinte forma:

```
17    delete [] vetor; // certo
18    delete vetor[]; //errado
19
```

Exercício

Faça um algoritmo que, utilizando alocação dinâmica, pesquise um número em uma lista (vetor). Receba do usuário o tamanho do vetor e o número a ser pesquisado. Utilize três funções, uma para preencher o vetor, outra para pesquisar e outra para exibir. Retorne para o usuário se o número foi encontrado ou não no vetor.