INSTITUTO DE INFORMÁTICA UFRGS

INF01040 – Introdução à Programação

Algoritmos Iterativos

Arranjos Unidimensionais (Vetores ou Arrays)

Uso de estruturas de tipo de dados indexada e homogêneas em C

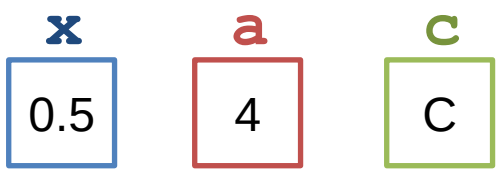
Recapitulando...

- Vimos até aqui como declarar **variáveis** de diversos tipos em C

`float` `int` `char`

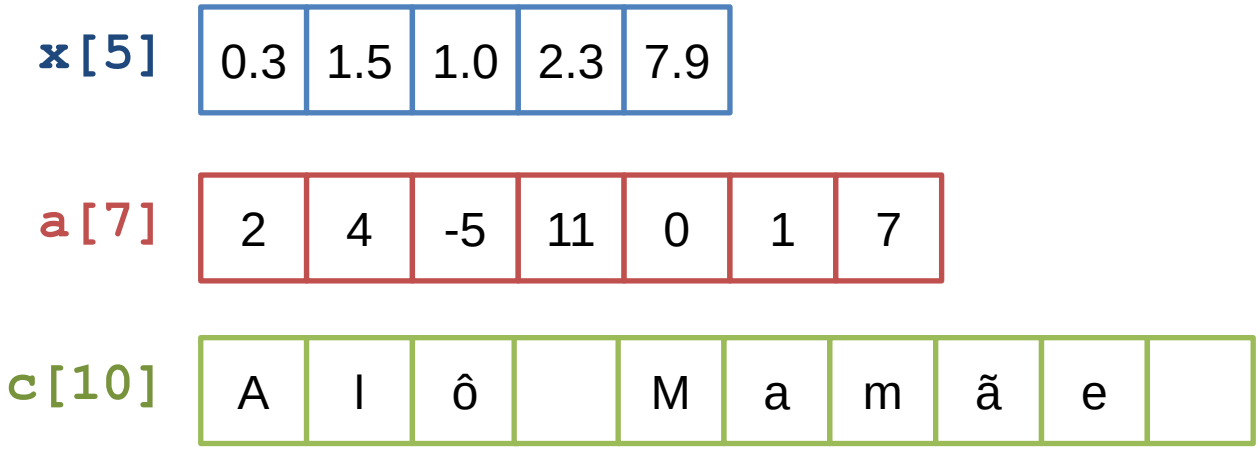
- Vimos também como cada variável de um tipo é capaz de armazenar **apenas um valor**

```
int a = 4;
float x = 0.5;
char c = 'C';
```



Nessa aula

- Vamos ver como podemos utilizar **arranjos** para definir e armazenar vários valores de um **mesmo tipo**



Considere o enunciado

- Ler 30 valores e calcular a média aritmética dos mesmos
- Análise do problema
  - *Quantas variáveis são necessárias para ler os 30 valores?*
    a) *30 variáveis?*
    b) *1 variável?*

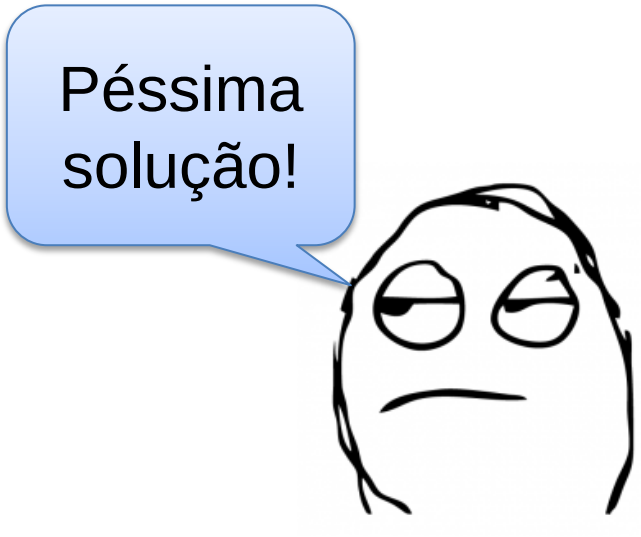
As soluções a e b são possíveis

***30 variáveis** diferentes ou **1 variável acumuladora** onde todos os valores lidos são somados para depois calcular a média*

Solução com 30 variáveis (Algoritmo)

```
// Lê 30 valores e calcula sua média aritmética
Variáveis:
    Inteiro: valor1, valor2, valor3, valor4,....,valor30
    Inteiro: soma
    Real:  media

Início
      Ler(valor1, valor2, valor3,....valor30)
      soma = valor1+valor2+valor3+...+valor30
      media = soma / 30
      Escrever(media)
Fim
```



Solução com 30 variáveis (Implementação em C)

```c
//Le trinta valores e calcula sua media aritmetica
#include <stdio.h>
int main()
{
   int valor1, valor2, valor3... valor30, soma;
   float media;
   printf("\nValor 1: ");
   scanf("%d", &valor1);
   printf("\nValor 2: ");
   scanf("%d", &valor2);
   printf("\nValor 3: ");
   scanf("%d", &valor3);
   (...)
   soma = valor1 + valor2 + valor3 + ... + valor30;
   media = (float)soma / 30;
   printf("\nMedia = %.2f\n", media);
}
```

Solução com 1 variável acumuladora (Algoritmo)

```
// Lê N valores e calcula sua média aritmética
Constantes: N = 30
Variáveis:
   Inteiro: valor, soma, i
     Real: media

Início
       soma = 0;
       Para (i=0; i<N; i++) {
           Ler (valor)
           soma = soma + valor
      }
      media = soma / N
      Escrever(media)
Fim
```

Solução com 1 variável acumuladora (Implementação em C)

```c
//Le trinta valores e calcula sua media aritmetica
#include <stdio.h>
#define MAX 30
int main()
{
  int i, valor, soma;
  float media;
  soma = 0;
  printf("Forneca %d valores (inteiros):\n", MAX);
  for(i=0;i<MAX;i++){
    printf("Valor %d: ",i);
    scanf("%d", &valor);
    soma = soma + valor;
  }
  media = (float)soma / MAX;
  printf("\nMedia = %.2f\n", media);
  return 0;
}
```

Mas, se o enunciado fosse...

- Ler 30 valores, calcular a média aritmética dos mesmos e **imprimir a lista de valores que ficarem acima da média**

- Análise do problema
  - *Quantas variáveis são necessárias para ler os 30 valores?*
    - a) ~~*30 variáveis?*~~
    - b) ~~*1 variável?*~~

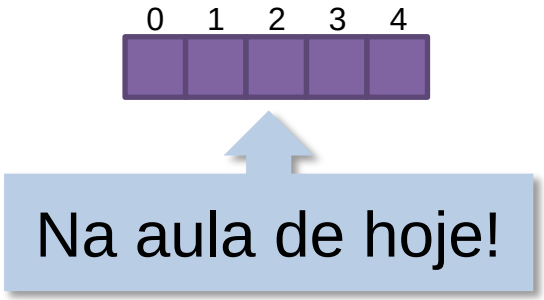
c) usar arranjos

Não é mais viável usar as soluções anteriores

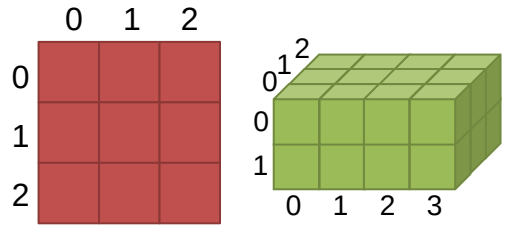
*com **1 variável acumuladora** se perdem os valores originalmente lidos e com **30 variáveis** já vimos que a solução não é das mais elegantes*

Arranjos

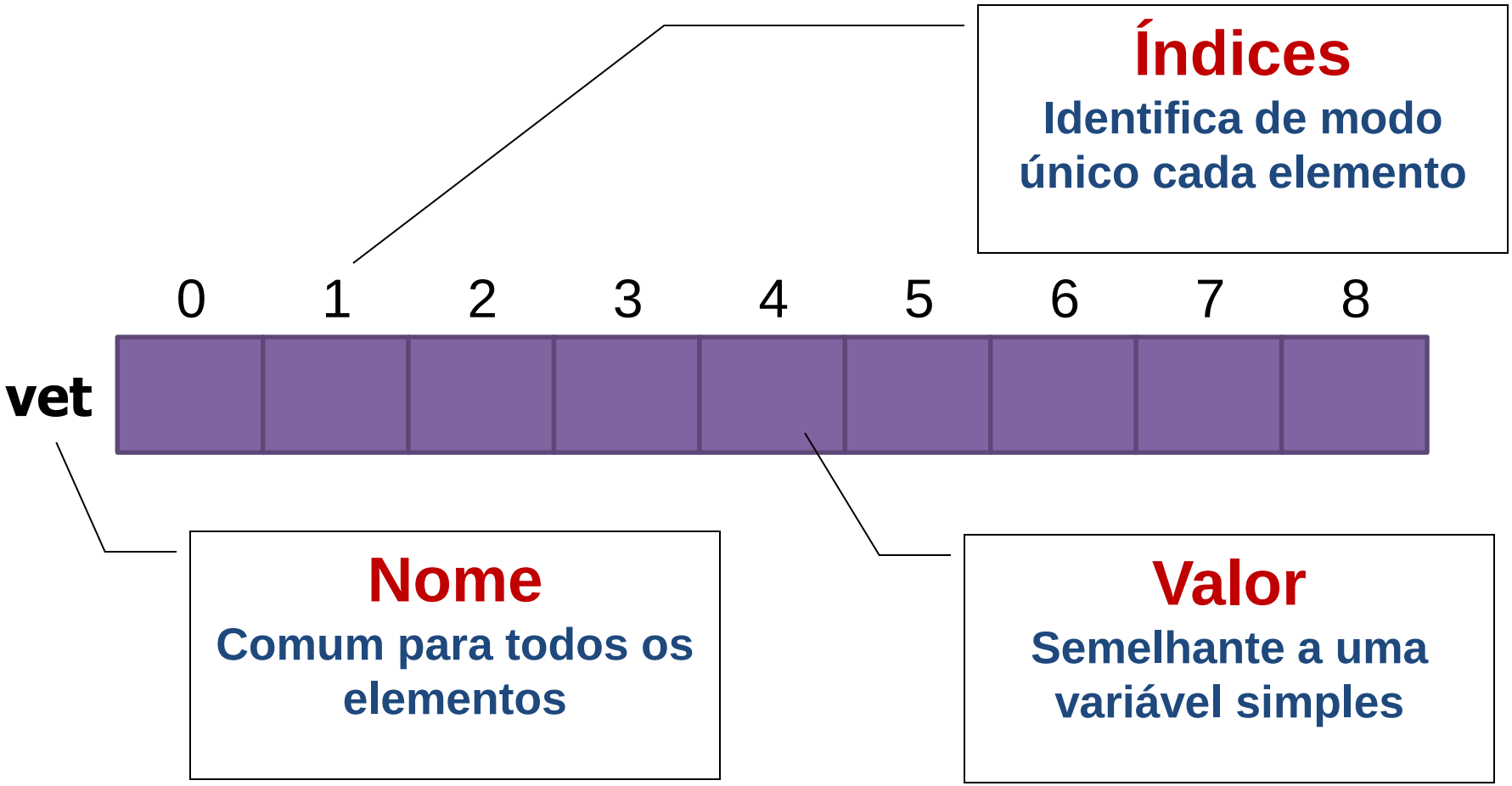
- Estruturas para armazenamento de múltiplos elementos de dados
  - Armazenamento de dados de **mesmo tipo**
  - Armazenamento **contíguo** na memória
  - Acesso **indexado**
- Unidimensionais
  - Vetores ou Arrays



- Multidimensionais
  - Matrizes, cubos, etc.



Arranjos Unidimensionais (Vetores ou Arrays)



Declaração de Vetores

Forma geral

```
tipo nome [tamanho];
```

Tipo de dado
(char, int, float, etc.)

Nome da variável
(Escolhido pelo programador)

Tamanho do Vetor
(Quantos valores são armazenados)

- **Exemplos:**
  - Info é um vetor real de 10 elementos, cujos índices podem variar de 0 a 9:

```
float info[10];
```

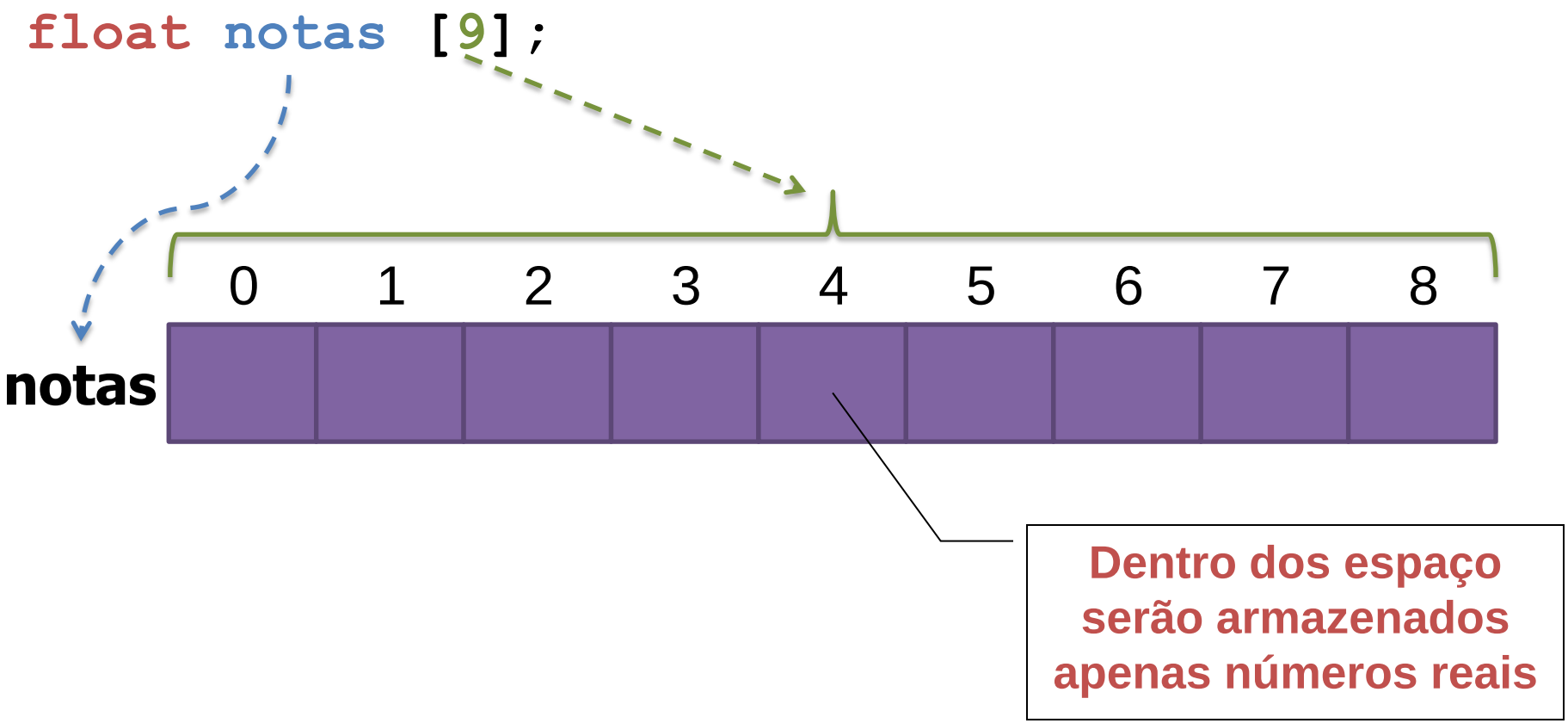
  - Notas é um vetor inteiro de 50 elementos, cujos índices podem variar de 0 a 49:

```
#define LIM_SUP 50
int notas[LIM_SUP];
```

  - Temperaturas é um vetor real de 101 elementos, cujos índices podem variar de 0 a 100:

```
#define LIM_TEMP 101
float temperaturas[LIM_TEMP];
```

Declarando um vetor de 9 notas de alunos



Inicialização de Vetores

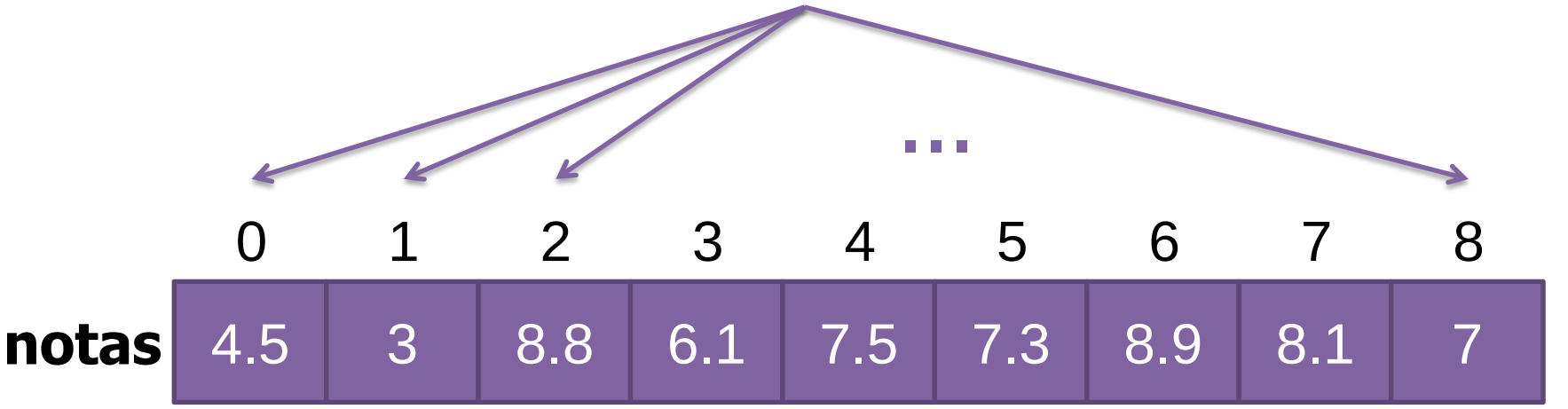
- Vetores, como as demais variáveis, ao serem criados **não limpam** a área da memória onde serão alocados os valores
- Vetores podem ser inicializados de 3 formas:
  1) **na declaração;**
  2) **por atribuição, em algum momento da execução;**
  3) **por leitura.**

1) Inicialização na declaração

Forma geral

```
tipo nome[tam] = {valor0, valor1, ... valortam-1};
```

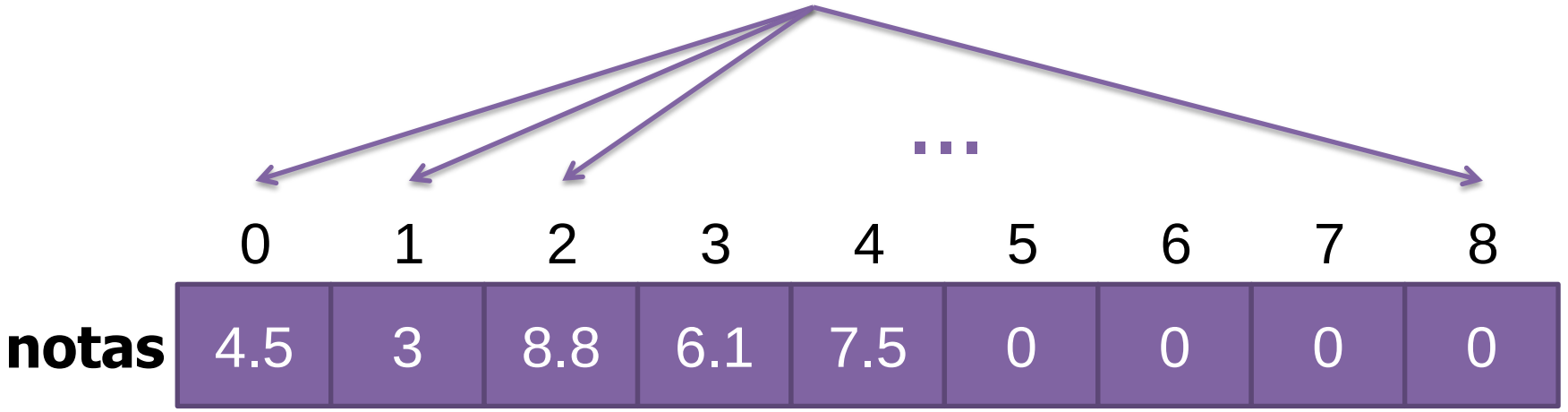
```
float notas [9] = {4.5, 3, 8.8, 6.1, 7.5, 7.3, 8.9, 8.1, 7};
```



1) Inicialização na declaração

Atenção: Posições não inicializadas são preenchidas com zero!

```
float notas [9] = {4.5, 3, 8.8, 6.1, 7.5};
```



2) Inicialização por atribuição

Inicializar um vetor na declaração assim...

```
int numeros[6] = {4, 3, 8, 1, 5, 3};
```

é o mesmo que:

```
int numeros[6];
numeros[0] = 4;
numeros[1] = 3;
numeros[2] = 8;
numeros[3] = 1;
numeros[4] = 5;
numeros[5] = 3;
```

Inicialização por atribuição, durante a execução

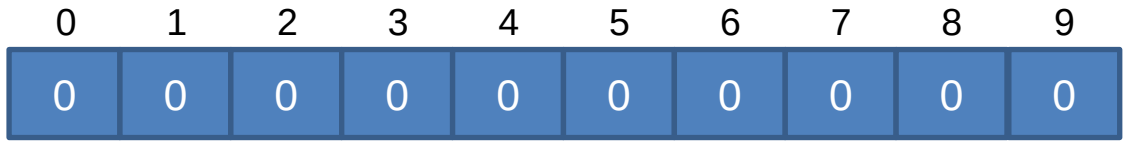
2) Inicialização por atribuição

Exemplo 1:

Zerar todo o conteúdo de um vetor de 10 posições

```
int vet[10];
for (i=0; i<10; i++)
  vet[i] = 0;
```

i varia de 0 a 9 iterando todos os índices do vetor

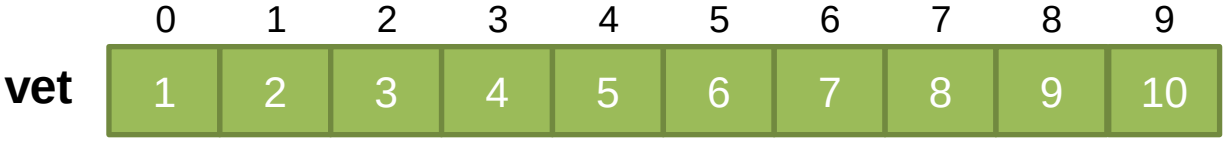


Exemplo 2:

Inicializar um vetor de 10 posições com valores consecutivos a partir de 1

```
int vet[10];
for (i=0; i<10; i++)
  vet[i] = i+1;
```

Aqui *i* é usado para atribuir valor dinamicamente ao vetor



3) Inicialização por leitura

- O usuário informa os valores que devem ser armazenados no vetor

Exemplo 1:

Ler 3 valores reais informados pelo usuário armazenando em um vetor

```
float reais[3];
scanf("%f", &reais[0]);
scanf("%f", &reais[1]);
scanf("%f", &reais[2]);
```

Lê normalmente considerando o tipo (%f nesse caso) e indicando o índice onde armazenar

Exemplo 2:

Inicializar um vetor de inteiros de 10 posições com valores informados pelo usuário

```
int inteiros[10];
for (i=0; i<10; i++)
  scanf("%d", &inteiros[i]);
```

Aqui usamos um for para evitar escrever 10 vezes scanf

Apresentação do conteúdo de um vetor

- Depois de declarar o vetor e atribuir (inicializar) valores nas suas posições podemos **apresentar** esses valores novamente ao usuário

Exemplo 1:

Imprimir as primeiras duas posições de um vetor de números reais (assumindo que ele já foi declarado e inicializado)

```
printf("%f %f" , reais[0], reais[1]);
```

Não esqueça, aqui é **sem o &**

Exemplo 2:

Imprimir um vetor de 10 posições de inteiros (assumindo que ele já foi declarado e inicializado)

```
for (i=0; i<10; i++)
  printf("%d ", inteiros[i]);
```

Note o espaço no printf para os números não saírem colados

Retomando o enunciado...

- Ler 30 valores, calcular a média aritmética dos mesmos e **imprimir a lista de valores que ficarem acima da média**
- Perguntas:
  - *Quantas posições precisa ter o nosso arranjo?*
  - *De que forma devemos iniciar os valores do arranjo?*
  - *De que forma imprimimos a lista?*

Retomando o enunciado...

- Ler 30 valores, calcular a média aritmética dos mesmos e **imprimir a lista de valores que ficarem acima da média**
- Perguntas:
  - *Quantas posições precisa ter o nosso arranjo?*
    - ***30 valores*** *(vamos assumir inteiros)*
  - *De que forma devemos iniciar os valores do arranjo?*
    - ***Por leitura****, os valores são informados pelo usuário*
  - *De que forma imprimimos a lista?*
    - *Percorrendo e comparando os valores com a média*

Solução Algorítmica

```
// Lê N valores, calcula sua média aritmética e imprime os valores lidos que estão acima da média
Constantes: N = 30
Variáveis:
   Inteiro: valor[N], soma, i
   Real: media

Início
       // le N valores e os armazena no vetor e já realiza o somatório dos mesmos
      soma = 0;
      Para (i=0; i<N; i++) {
           Ler (valor[i])
           soma = soma + valor[i]
      }
      // calcula a média
      media = soma / N
      Escrever(media)
      // encontra e imprime valores acima da média
      Para (i=0; i<N; i++) {
          Se (valor[i] > media)
              Escrever(valor[i])
      }
Fim
```

Solução em C

```c
#include <stdio.h>
#define MAX 30
int main()
{
    int vetor[MAX];
    int i, soma = 0;
    float media;
    //Preenche o vetor e já realiza a soma dos valores
    printf("Informe %d valores inteiros: ", MAX);
    for(i=0; i<MAX; i++){
        scanf("%d", &vetor[i]);
        soma = soma + vetor[i];
    }
    //Calcula a media
    media = (float)soma/MAX;
    printf("Media: %.2f\n", media);
    //Lista os valores do vetor acima da media
    for(i=0; i<MAX; i++){
        if(vetor[i] > media)
            printf("%d ", vetor[i]);
    }
    return 0;
}
```

Lembretes importantes!!!

- O índice da primeira posição de um vetor em Linguagem de Programação C é **zero**

Ex.: `num[0] = 10;`

- O sistema não controla a correção dos índices usados
  - Quem deve garantir que os índices estejam dentro do intervalo correto é você, sr. programador!!!

```
int vet[5] = {1, 2, 3, 4, 5};
printf("%d", vet[5]);
```

Isso é um erro e o compilador não acusa, o resultado aqui é incerto

Lembretes importantes!!!

- Os índices são sempre contínuos, inteiros e positivos
  - Por exemplo, **não** é possível declarar um vetor só com os índices pares
  - **Nem** com caracteres ou valores reais como índices
- Não se pode misturar valores de tipos diferentes em vetores
  - Um vetor de inteiros não vai funcionar para armazenamento de reais ou caracteres

Pseudo-código e Fluxogramas

- Vimos nessa aula a declaração e uso de arranjos direto usando a sintaxe da linguagem C
- Até agora não estávamos declarando variáveis em Pseudo-código, apenas as usávamos direto
- Para declarar um arranjo em pseudo-código ou fluxograma vamos usar a mesma sintaxe do C, apenas trocando os tipos para português

Linguagem C		**Pseudo-código**
`int vet[5] = {9, 8, 7, 0, 1};`	→	`Inteiro vet[5] = {9, 8, 7, 0, 1};`
`float arr[3] = {0, 0.3, 0.6};`	→	`Real arr[3] = {0, 0.3, 0.6};`
`char nome[10] = "INF01040";`	→	`Caractere nome[10] = "INF01040";`