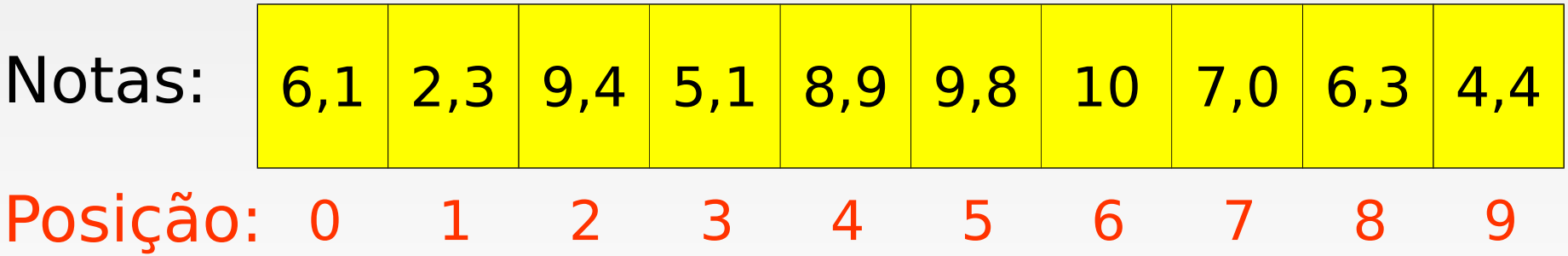
Arranjos unidimensionais

- Correspondem a posições da memória:
  - identificadas por um **único nome**
  - individualizadas por **índices**
  - cujo conteúdo é de um **mesmo tipo**



Arranjos unidimensionais

- Utilizados para armazenar conjuntos de dados cujos elementos podem ser endereçados por um único índice.

- Também são conhecidos como vetores.

Vetores na Linguagem C

- Como declarar:

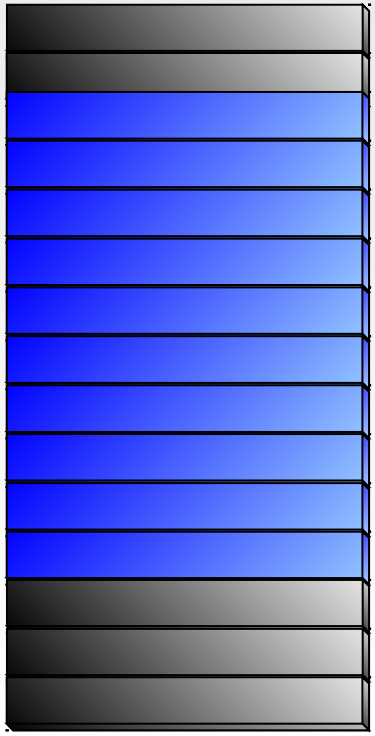
```
<tipo> <nome> [<tamanho1>][<tamanho2>]...;
```

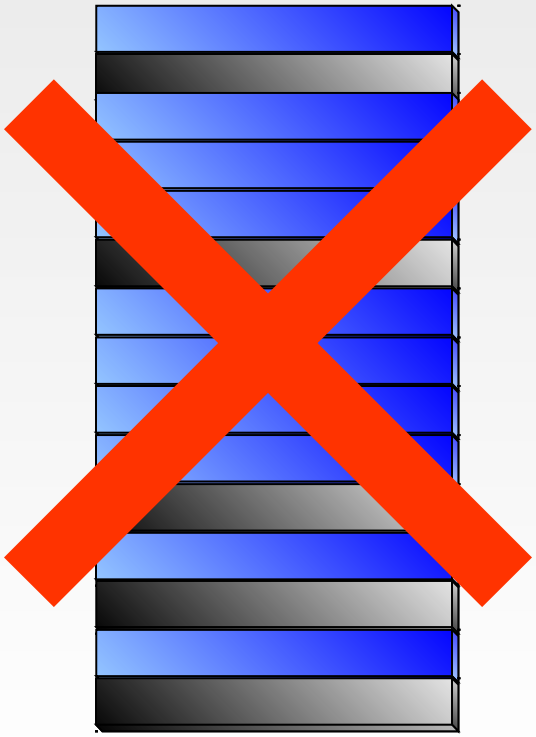
- Exemplos:

```
float VetReais[100];
int   Vetor[5][9];
char  Nome_cliente[50];
float cubo[20][12][7];
```

Vetores na Linguagem C

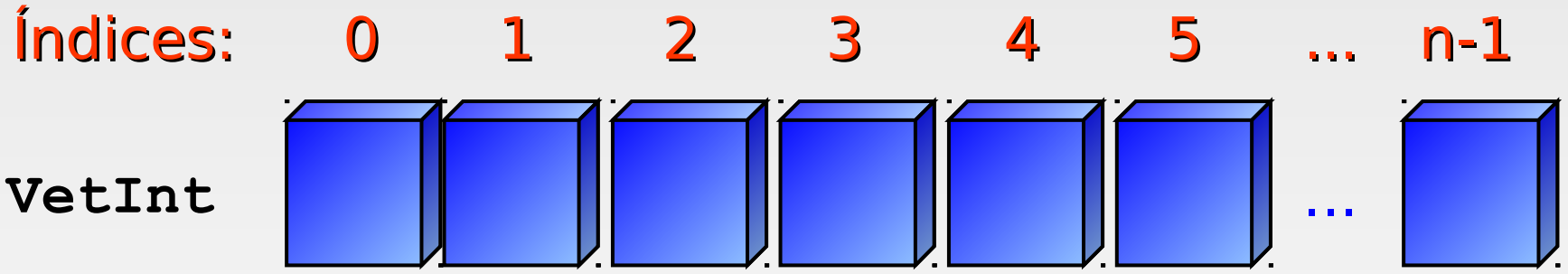
- O compilador C aloca uma porção contígua da memória para armazenar os elementos das matrizes e vetores.





Vetores na Linguagem C

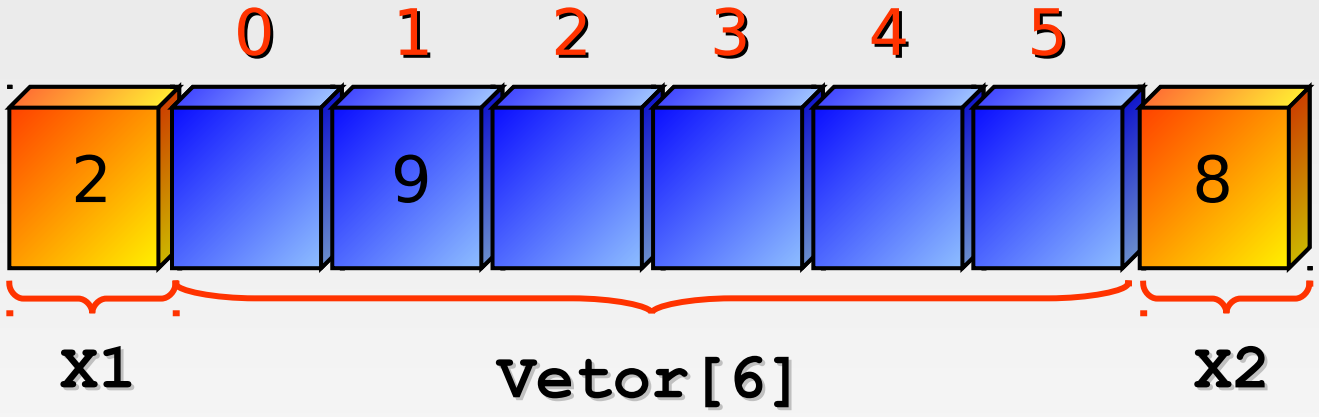
```
int VetInt[n];
```



Índice do primeiro elemento: zero
Índice do último elemento: n – 1
Quantidade de elementos: n

Vetores e Matrizes na Linguagem C

- Índices fora dos limites podem causar comportamento anômalo do código.




```
int X1;
int Vetor[6];
int X2
```

```
Vetor[1]  = 9;
Vetor[-1] = 2;
Vetor[6]  = 8;
```

Vetores e Matrizes na Linguagem C

- O tamanho de um vetor ou matriz é **pré-definido**, ou seja, após a compilação, não pode ser mudado.

- Portanto, vetores e matrizes são chamadas **estruturas de dados estáticas**, pois mantém o **mesmo tamanho** ao longo de toda a execução do programa.

Vetores e Matrizes na Linguagem C :: Exemplos

- Atribuir valores na declaração do vetor:

```
int vetor[5] = {1,2,3,4,5};
```

Vetores e Matrizes na Linguagem C :: Exemplos

- Colocar os números de 1 a 5 num vetor:

```
for (i=0; i<5; i++)
        Vetor[i] = i + 1;
```

- Colocar os números de 5 a 1 num vetor:

```
for (i=0; i<5; i++)
        Vetor[i] = 5 - i;
```

Vetores e Matrizes na Linguagem C :: Exemplos

- Copiar dados de um vetor para outro:

```
#define TAM_MAX 10
double VetReais[TAM_MAX], VetCopia[TAM_MAX];
for (i=0; i<TAM_MAX; i++)
        VetCopia[i] = VetReais[i];
```

- **Boa prática de programação:**
  - Definir o tamanho de vetores com **constantes** flexibiliza a manutenção do código.

Vetores e Matrizes na Linguagem C :: Exemplos

- Leitura dos dados de um vetor:

```
for (i=0; i<TAM_MAX; i++)
{
        printf("Digite um número: ");
         scanf("%f", &Vet[i]);
}
```

Vetores e Matrizes na Linguagem C :: Problema B

- Ler um elemento K.
- Ler um vetor A de N elementos.
- Verificar se o elemento K está presente no vetor:
  - Se estiver, imprimir a posição onde ele foi encontrado.
  - Caso contrário, imprimir mensagem "elemento K não encontrado".

Vetores e Matrizes na Linguagem C :: Problema C

- Ler N valores inteiros (N ≤ 100) até que seja digitado o valor zero.
- A seguir, inverter o vetor, trocando o 1º elemento com o último, o 2º com o penúltimo, e assim sucessivamente.
- Ao final, imprimir o vetor invertido.

Vetores e Matrizes na Linguagem C :: Problema D

- Faça um programa que imprima uma matriz quadrada de dimensão N contendo:
  - o número 1 nos elementos abaixo da diagonal principal
  - o número 0 nos demais elementos

- N deve ser menor ou igual a 20.

Exemplos

1. Armazenar 10 valores inteiros num vetor de 10 posições e mostrar os valores armazenados.
2. Armazenar 10 valores inteiros num vetor de 10 posições. Após, leia o vetor e mostre os valores armazenados, adicionando em 10 unidades quando forem números positivos.
3. Faça um algoritmo que leia 5 idades de 5 pessoas e escreva o número de pessoas que são maiores de idade. Armazene as idades num vetor.
4. Faça um programa que armazene em um vetor de inteiros as quantidades compradas de 5 produtos. Em outro vetor de reais, armazene o valor unitário de cada produto. O programa deve, ao final, mostrar o valor total a ser pago por cada produto. Considere que o índice do vetor corresponde ao código do produto.

1. Armazenar 10 valores inteiros num vetor de 10 posições e mostrar os valores armazenados.

```
#include<stdio.h>
#include<stdlib.h>

#define QUANT 10

int main(){
	int vetor[QUANT], i;

	for (i=0; i<QUANT; i++) {
		printf("\nDigite um valor inteiro: ");
		scanf("%d", &vetor[i]);
	}

	for (i=0; i<QUANT; i++) {
		printf("\nO valor da posicao %d eh %d\n", i, vetor[i]);
	}

	return 0;
}
```

2. Armazenar 10 valores inteiros num vetor de 10 posições. Após, leia o vetor e mostre os valores armazenados, adicionando em 10 unidades quando forem números positivos.

```
#include<stdio.h>
#include<stdlib.h>

#define QUANT 10

int main() {
	int vetor[QUANT], cont;

	for (cont=0; cont<QUANT; cont++) {
		printf ("\nDigite um valor inteiro: ");
		scanf ("%d", &vetor[cont]);
	}

	for (cont=0; cont<QUANT; cont++) {
		if (vetor[cont]>0){
			printf ("\nO valor na posicao %d eh %d\n", cont, vetor[cont]+10);
		} else {
		printf ("\nO valor na posicao %d eh %d\n", cont, vetor[cont]);
		}
	}
	return 0;
}
```

3. Faça um algoritmo que leia 5 idades de 5 pessoas e escreva o número de pessoas que são maiores de idade. Armazene as idades num vetor.

```
#include<stdio.h>
#include<stdlib.h>

#define QUANT 5

int main() {
        int pessoas[QUANT];
        int i=0, numMaiorIdade=0;

        while (i<QUANT) {
                printf ("\nDigite uma idade: ");
                scanf ("%d", &pessoas[i]);
                if (pessoas[i]>=18){
                        numMaiorIdade++;
                }
        i++;
        }

   printf ("\nO numero de pessoas maiores eh %d\n", numMaiorIdade);

        return 0;
}
```

Vetor não tem muita utilidade nesse problema

4. Faça um programa que armazene em um vetor de inteiros as quantidades compradas de 5 produtos. Em outro vetor de reais, armazene o valor unitário de cada produto. O programa deve, ao final, mostrar o valor total a ser pago por cada produto. Considere que o índice do vetor corresponde ao código do produto.

```
#include<stdio.h>
#include<stdlib.h>
#define NUMPROD 5

int main(){
        int i;
        int quantidade[NUMPROD];
        float precoUnitario[NUMPROD];

        for (i=0; i<NUMPROD; i++) {
                printf ("\nDigite a qtidade comprada do produto %d ", i);
                scanf ("%d", &quantidade[i]);
                printf ("\nDigite o preco unitario do produto %d ", i);
                scanf ("%f", &precoUnitario[i]);
        }
        for (i=0; i<NUMPROD; i++) {
          printf ("\nO valor total da compra %d eh %.2f\n", i,
                                quantidade[i]*precoUnitario[i]);
        }
        return 0;
}
```