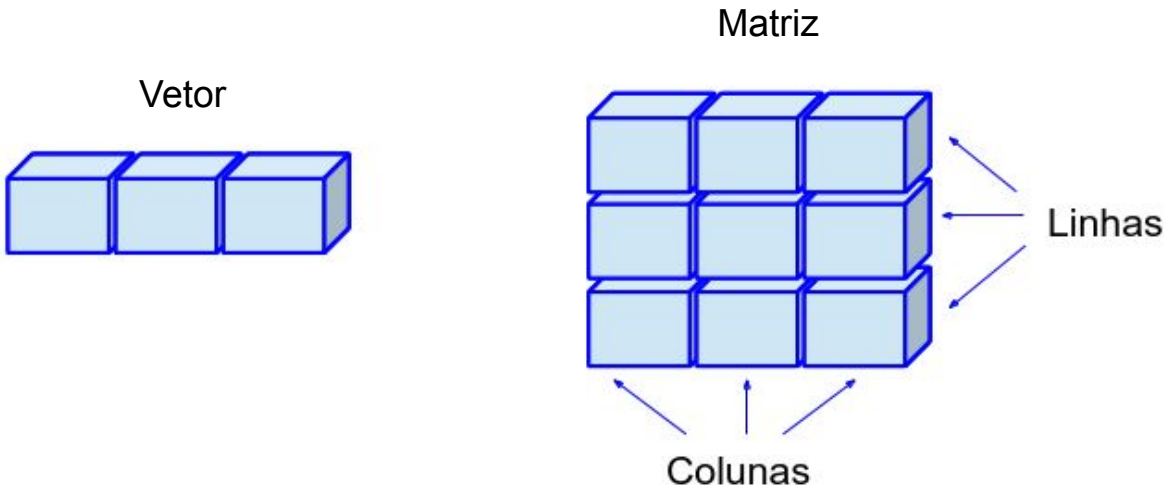
Introdução à linguagem C
Aula 03 - Vetor e Matriz

Professor: Racyus Delano
E-mail: racyuspacifico@univicosa.com.br

Vetores e matrizes

- **Uma coleção de dados**
  - Mesmo nome;
  - Mesmo tipo;
  - Armazenados em um bloco contínuo de memória;
  - Evita declaração de várias variáveis.



Vetores

- **Declarando vetor**
  - **Formato:**
    - **<Tipo de dados> <nome da variável> [<tamanho>];**
  - **Declaração:**
    - Aloca um espaço de memória contínuo;
    - Estático: tem o mesmo tamanho após a declaração.
  - **Exemplos:**

```
int c[12];
int scores[300];
float weight[3284];
char alphabet[26];
```

Type is int.
Name is c.

Vetores

- **Definindo uma constante como o tamanho vetor**
  - Use uma constante para definir o tamanho do vetor.
    - Melhora a legibilidade;
    - Melhora a versatilidade;
    - Melhora a manutenção.

  - **Exemplos:**

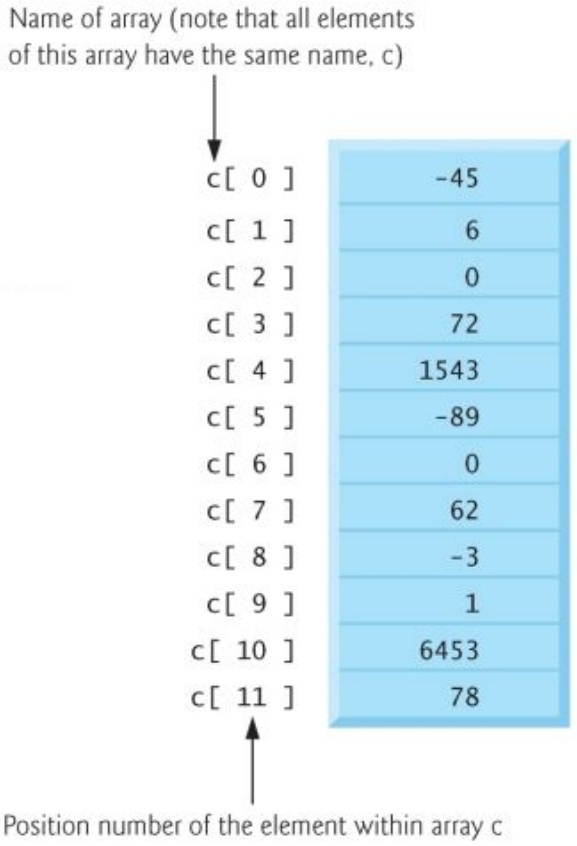
```
const int NUMBER_OF_STUDENTS = 50;
// ..
int scores[NUMBER_OF_STUDENTS];
```

```
#define NUMBER_OF_STUDENTS 50
    // ..
    int scores[NUMBER_OF_STUDENTS];
```

Vetores

- **Acessando elementos do vetor**
  - Os elementos do vetor são acessados através do índice (*index*);
    - **Formato: <nome do vetor> [índice]**
  - O primeiro elemento do vetor é armazenado no índice 0.
  - **Exemplo:**

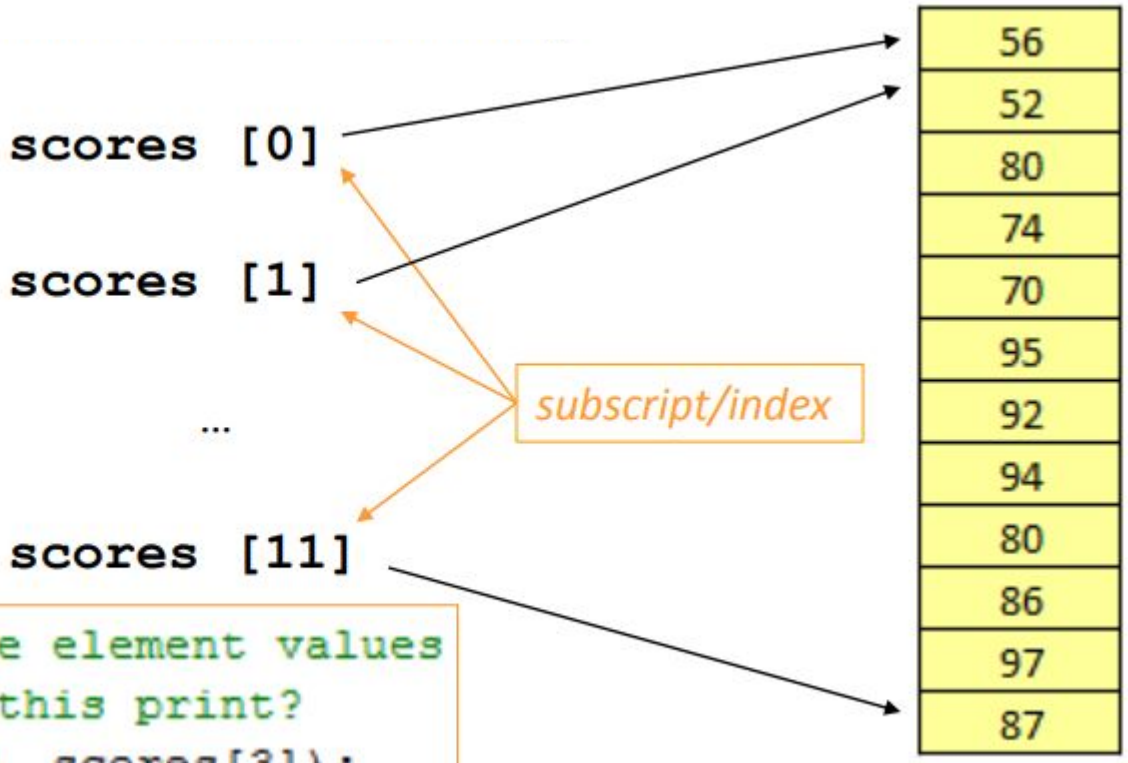
```
printf("%d\n", c[5]);
```



Vetores

- **Acessando elementos do vetor**
  - **Exemplo:** Analise a figura do vetor scores. Qual é o elemento da posição 3?

```
int scores[12];
```

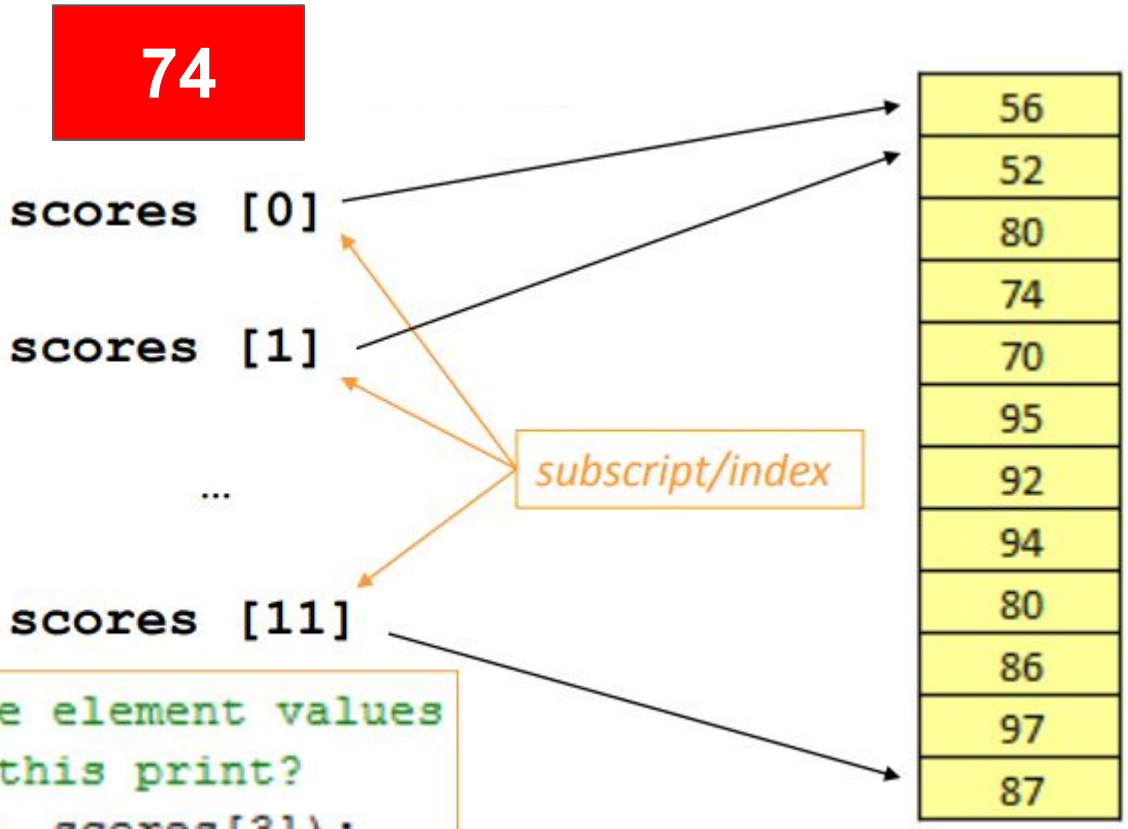



```
// Given these element values
// What does this print?
printf("%d\n", scores[3]);
```

Vetores

- **Acessando elementos do vetor**
  - **Exemplo:** Analise a figura do vetor scores. Qual é o elemento da posição 3?

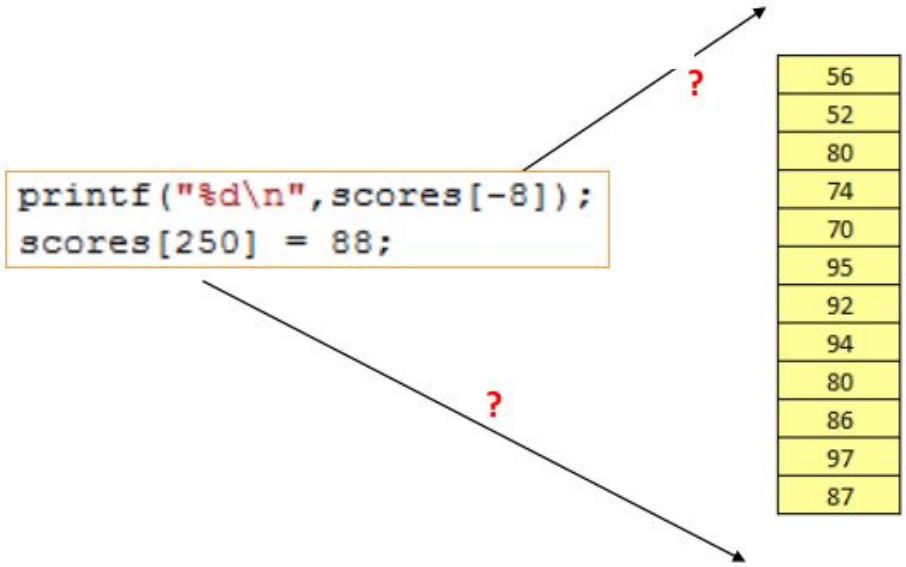
```
int scores[12];
```




```
// Given these element values
// What does this print?
printf("%d\n", scores[3]);
```

Vetores

- **Armadilha acesso elementos**
  - Índices do vetor são de 0 a tamanho-1;
  - C permite você sair fora dos limites do vetor;
    - Resultados imprevisíveis que geram falhas de segmentação;
    - O compilador não irá detectar esses erros;
  - O programador é o responsável em verificar.



Vetores

- **For com vetores - Exemplo**

```
int scoreSub;
// Print forward
for (scoreSub = 0; scoreSub < 12; scoreSub++)
   printf("Score %d is %d\n", scoreSub+1,
             scores[scoreSub]);
// Print backward, in reverse
for (scoreSub = 11; scoreSub >= 0; scoreSub--)
   printf("Score %d is %d\n", scoreSub+1,
             scores[scoreSub]);
```

56
52
80
74
70
95
92
94
80
86
97
87

```
Score 1 is 56
Score 2 is 52
Score 3 is 80
Score 4 is 74
...
Score 12 is 87
```

```
Score 12 is 87
Score 11 is 97
Score 10 is 86
Score 9 is 80
...
Score 1 is 56
```

Vetores

- **Uso de constantes**
  - Pode ser usado em todos os lugares onde o tamanho do vetor é referenciado.
    - For

```
int score;
for (score=0; score<NUMBER_OF_STUDENTS; score++)
   printf("%d\n", scores[score]);
```

    - Em cálculos envolvendo o tamanho

```
lastIndex = NUMBER_OF_STUDENTS - 1;
lastScore = scores[NUMBER_OF_STUDENTS - 1];
```

    - Passando um vetor por função

```
total = sum_scores(scores, NUMBER_OF_STUDENTS);
```

Vetores

- **Vetor como parâmetro de função**
  - Inclui o tipo e colchetes [];
    - O tamanho entre colchetes é opcional e ignorado;
  - Passa o ponteiro/referência para o vetor;
    - A função pode modificar os elementos do vetor;
  - É comum passar o tamanho.
  - **Exemplo:**

```
void print_scores(int values[], int num_values) {
   // Call: print_scores(scores, scoreCount)
   int valueNdx;
   for (valueNdx=0; valueNdx<num_values; valueNdx++)
      printf("%d\n", values[valueNdx]);
}
```

Vetores

- **Inicializando o vetor**
  - Vetores podem ser inicializados na declaração;

```
int scores[3] = {76, 98, 83};
```

    - O tamanho não pode ser uma variável ou constante;
  - Equivalente a:

```
int scores[3];
scores[0] = 76;
scores[1] = 98;
scores[2] = 83;
```

Vetores

- **Inicializando o vetor**
  - Os elementos não inicializados recebem valor 0;

```
int scores[5] = {76, 98, 83}
```

  - **Exemplo:**

```
scores[0] = 76;
scores[1] = 98;
scores[2] = 83;
scores[3] = 0;
scores[4] = 0;
```

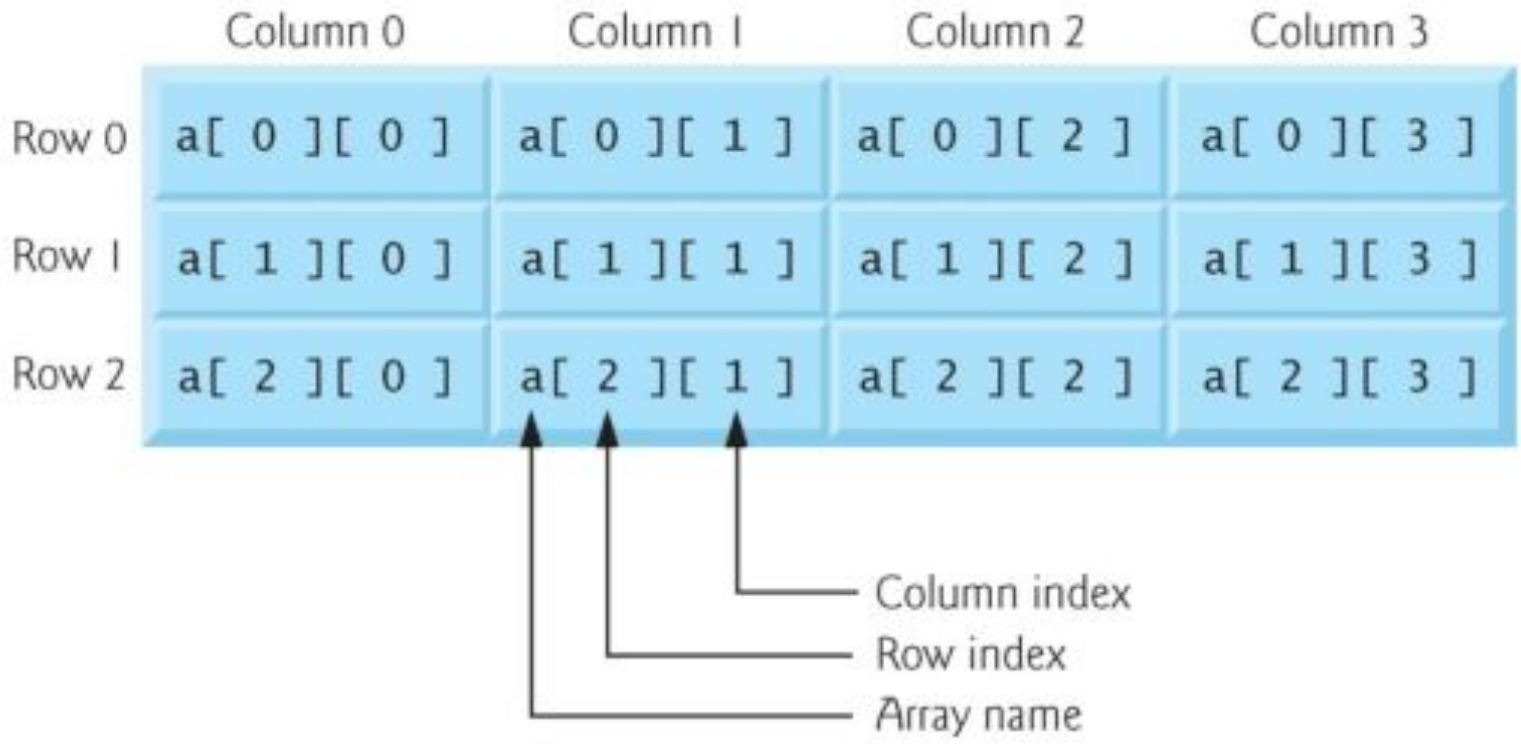
Vetores

- **Inicializando o vetor**
  - Se o tamanho do vetor não for atribuído
    - O tamanho do vetor será o número de elementos recebidos na inicialização.
  - **Exemplo:**

```
int scores[] = {76, 98, 83}
```

Matrizes

- Tem duas dimensões (Linha e Coluna);
  - **<u>Exemplo:</u> int a[3][4];**



Matrizes

- **Inicializando matrizes**
  - Valores não especificados recebem 0;
    - **Exemplo:**

```
int nums[4][5] = { {10,  6, -7, 13, 28},
                   {10,  5, 44,  8},
                   {33, 20,  1,  0, 14},
                   { 2, 66, 25, 37,  1}
                 }
```

Matrizes

- **For com matrizes - Exemplo**

```
int nums[4][5] = { {10,  6, -7, 13, 28},
                   {10,  5, 44,  8},
                   {33, 20,  1,  0, 14},
                   { 2, 66, 25, 37,  1}
                 }
```

```
//Imprimindo elementos da matriz
int i, j, nums;

for(i=0; i<4; i++)
{
  for(j=0; j<5; j++)
  {
     printf("%d", nums[i][j]);
  }
  printf("\n");
}
```

Exercícios sala de aula

1. Desenvolva um algoritmo que leia 2 vetores de 5 elementos inteiros. Em seguida, calcule a soma desses vetores, guarde o resultado em um terceiro vetor e escreva o resultado.

Exercícios sala de aula

1. Desenvolva um algoritmo que leia 2 vetores de 5 elementos inteiros. Em seguida, calcule a soma desses vetores, guarde o resultado em um terceiro vetor e escreva o resultado.

```c
#include <stdio.h>

int main()
{
    int i, N = 5;
    int vet1[N], vet2[N], soma[N];

    printf("Armazenando elementos vetores\n");
    for(i=0; i<N; i++)
    {
      printf("Digite os elementos do vet1: ");
      scanf("%d", &vet1[i]);
      printf("Digite os elementos do vet2: ");
      scanf("%d", &vet2[i]);
      printf("\n");
    }

    printf("Somando elementos vetores vet1 e vet2");
    for(i=0; i<N; i++)
      soma[i] = vet1[i] + vet2[i];

    printf("\nVetor vet1:");
    for(i=0; i<N; i++)
      printf("%d ", vet1[i]);

    printf("\nVetor vet2:");
    for(i=0; i<N; i++)
      printf("%d ", vet2[i]);

    printf("\nVetor soma:");
    for(i=0; i<N; i++)
      printf("%d ", soma[i]);

    return 0;
}
```

Exercícios sala de aula

2. Desenvolva um algoritmo que leia uma matriz 4x4 e imprima os elementos da sua diagonal principal e diagonal secundária.

Exercícios sala de aula

2. Desenvolva um algoritmo que leia uma matriz 4x4 e imprima os elementos da sua diagonal principal e diagonal secundária.

```c
#include <stdio.h>

int main()
{
    int i, j, N = 4;
    int mat[N][N];

    printf("Armazenando elementos na matriz\n");
    for(i=0; i<N; i++)
    {
      for(j=0; j<N; j++)
      {
        printf("Digite os elementos da matriz: ");
        scanf("%d", &mat[i][j]);
      }
      printf("\n");
    }

    printf("\nImprimindo elementos da matriz\n");
    for(i=0; i<N; i++)
    {
      for(j=0; j<N; j++)
      {
        printf("%d ", mat[i][j]);
      }
      printf("\n");
    }

    printf("\nImprimindo elementos diagonal principal\n");
    for(i=0; i<N; i++)
      printf("%d ", mat[i][i]);

    printf("\n\nImprimindo elementos diagonal secundária\n");
    for(i=0; i<N; i++)
      printf("%d ", mat[i][(N-1)-i]);

    return 0;
}
```

Exercícios da lista 01

- **Data de entrega: 29/02/2024.**

1. Escreva um programa que leia doze números do tipo inteiro digitados pelo usuário. Separe esses números em pares e ímpares, os armazenem em dois outros vetores chamados vetpar e vetimpar. Em seguida, o programa deve apresentar os resultados na tela.

2. Desenvolva um algoritmo que leia uma matriz numérica 4x4 e calcule:
   a. A soma dos elementos da diagonal secundária;
   b. A soma das linhas pares da matriz;
   c. A soma das linhas ímpares da matriz;
   d. A média das linhas pares;
   e. A média das linhas diagonais (primária e secundária).