Variáveis Indexadas

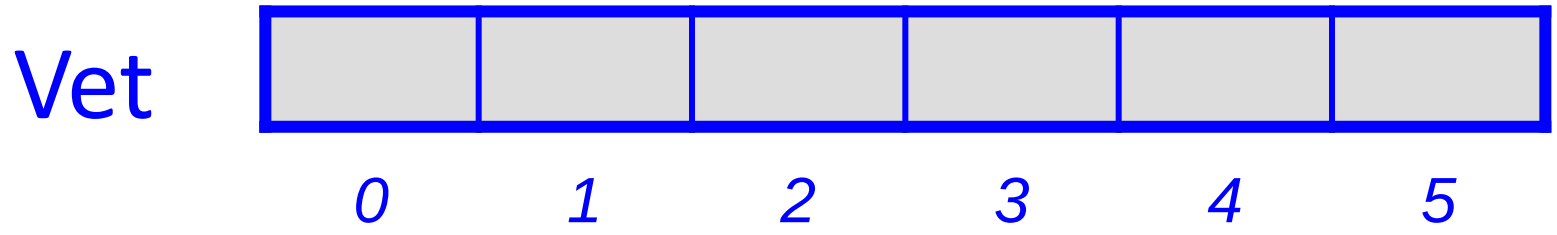
ECM404

Variáveis Indexadas – Array

- **Array** ou **“vetor”** é a forma mais familiar de dados estruturados.
- Um array é uma sequência de elementos do mesmo tipo, onde cada elemento é identificado por um índice.
  - A ideia é bem simples, criar um conjunto de variáveis do mesmo tipo utilizando apenas um nome.

Variáveis Indexadas – Array

- Os índices **sempre** começam em zero!



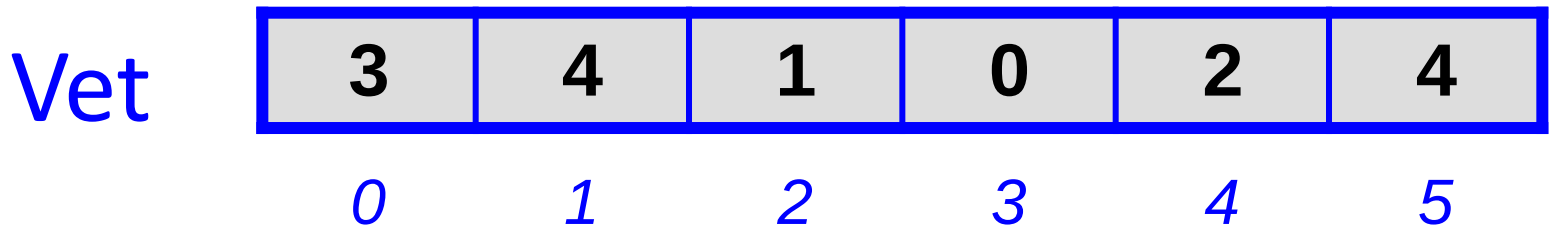
- Declaração do array:

```
int main (int argc, char *argv[]){
  int Vet[6];
  return 0;
}
```

Variáveis Indexadas – Array

- A declaração pode armazenar valores iniciais. Exemplo:

```
int main (int argc, char *argv[]){
  int Vet[6] = {3,4,1,0,2,4};
  return 0;
}
```



- **CUIDADO:** essa atribuição funciona apenas na declaração da variável!

Array – Exibição

- Para acessar os valores de um array, utilizaremos os índices:

```c
int main (int argc, char *argv[]){
  int Vet[6] = {3,4,1,0,2,4};
  int i;

  for(i=0;i<6;i++){
    printf("Vet[%i] = %i\n", i, Vet[i]);
  }

  return 0;
}
```

```
Vet[0] = 3
Vet[1] = 4
Vet[2] = 1
Vet[3] = 0
Vet[4] = 2
Vet[5] = 4
```

Array – Leitura e Exibição

```c
int main (int argc, char *argv[]){
  int Vet[6];
  int i;

  printf("***** LEITURA *****\n");
  for(i=0;i<6;i++){
    printf("Vet[%i] = ",i);
    scanf("%i",&Vet[i]);
  }

  printf("\n***** EXIBICAO *****\n");
  for(i=0;i<6;i++){
    printf("Vet[%i] = %i\n", i, Vet[i]);
  }

  return 0;
}
```

```
***** LEITURA *****
Vet[0] = 5
Vet[1] = 4
Vet[2] = 6
Vet[3] = 9
Vet[4] = 8
Vet[5] = 1

***** EXIBICAO *****
Vet[0] = 5
Vet[1] = 4
Vet[2] = 6
Vet[3] = 9
Vet[4] = 8
Vet[5] = 1
```

Array – Leitura e Exibição

```
int main (int argc, char *argv[]){
  int Vet[6];
  int i;

  printf("***** LEITURA *****\n");
  for(i=0;i<6;i++){
    printf("Vet[%i] = ",i);
    scanf("%i",&Vet[i]);
  }

  printf("\n***** EXIBICAO *****\n");
  for(i=0;i<6;i++){
    printf("Vet[%i] = %i\n", i, Vet[i]);
  }

  return 0;
}
```

```
***** LEITURA *****
Vet[0] = 5
Vet[1] = 4
Vet[2] = 6
Vet[3] = 9
Vet[4] = 8
Vet[5] = 1

***** EXIBICAO *****
Vet[0] = 5
Vet[1] = 4
Vet[2] = 6
Vet[3] = 9
Vet[4] = 8
Vet[5] = 1
```

Array – Leitura e Exibição

```c
#define N 6

int main (int argc, char *argv[]){
  int Vet[N];
  int i;

  printf("***** LEITURA *****\n");
  for(i=0;i<N;i++){
    printf("Vet[%i] = ",i);
    scanf("%i",&Vet[i]);
  }

  printf("\n***** EXIBICAO *****\n");
  for(i=0;i<N;i++){
    printf("Vet[%i] = %i\n", i, Vet[i]);
  }

  return 0;
}
```

```
***** LEITURA *****
Vet[0] = 5
Vet[1] = 4
Vet[2] = 6
Vet[3] = 9
Vet[4] = 8
Vet[5] = 1

***** EXIBICAO *****
Vet[0] = 5
Vet[1] = 4
Vet[2] = 6
Vet[3] = 9
Vet[4] = 8
Vet[5] = 1
```

Array – Expressões

- Cada elemento do array tem todas as características de uma variável e pode aparecer em expressões e atribuições (respeitando as mesmas regras das variáveis).

```
Vet[2] = 1;

valores[100] = 5.6789;

for(i=0;i<N;i++){
  somaNotas += notas[i]; // somaNotas = somaNotas + notas[i];
}
```

Array – Expressões

- Copiando um array:

```c
#include <stdio.h>
#define N 8

int main (int argc, char *argv[]){
  int V1[N] = {1,2,3,4,5,6,7,8};
  int V2[N];

  V2 = V1; // errado!

  int i;
  for(i=0;i<N;i++){
    V2[i] = V1[i];
  }

  return 0;
}
```

Array – Exercício

- Crie um programa que leia as notas de uma turma de 8 alunos e depois imprima as notas que são maiores que a média da turma.

Array – Exercício

```c
#include <stdio.h>
#define N 8

int main (int argc, char *argv[]){
  float notas[N];
  float somaNotas = 0, media;
  int i;

  printf("***** LEITURA *****\n");
  for(i=0;i<N;i++){
    printf("notas[%i] = ",i);
    scanf("%f",&notas[i]);
    somaNotas += notas[i];
  }
  media = somaNotas / N;
  printf("\nMedia da turma: %.1f\n", media);

  printf("\n***** EXIBICAO *****\n");
  for(i=0;i<N;i++){
    if(notas[i] >= media){
      printf("notas[%i] = %.1f\n", i, notas[i]);
    }
  }

  return 0;
}
```

```
***** LEITURA *****
notas[0] = 6
notas[1] = 7.8
notas[2] = 2.5
notas[3] = 3.4
notas[4] = 5
notas[5] = 9.5
notas[6] = 5.5
notas[7] = 7

Media da turma: 5.8

***** EXIBICAO *****
notas[0] = 6.0
notas[1] = 7.8
notas[5] = 9.5
notas[7] = 7.0
```

Array – Funções

- Para passar um array como parâmetro, basta utilizar o nome do array.

```
#include <stdio.h>
#define N 8

void lerVetor( int []);
int Maximo(int []);

int main (int argc, char *argv[]){
  int Vet[N];
  int max;

  lerVetor(Vet);
  max = Maximo(Vet);
  printf("Maior valor Vet[%i]: %i\n", max, Vet[max]);
  return 0;
}
```

Array – Protótipos

- Nos protótipos, não é necessário informar o tamanho do vetor.

```c
#include <stdio.h>
#define N 8

void lerVetor( int v[]);
int Maximo(int v[]);

int main (int argc, char *argv[]){
  int Vet[N];
  int max;

  lerVetor(Vet);
  max = Maximo(Vet);
  printf("Maior valor Vet[%i]: %i\n", max, Vet[max]);
  return 0;
}
```

Array – Definição da Função

- Definição da função lerVetor.

```
void lerVetor(int v[]){
  int i;
  for(i=0;i<N;i++){
    printf("v[%i] = ",i);
    scanf("%i",&v[i]);
  }
}
```

- Toda variável indexada é **passada por referência:** se a função **alterar** a variável indexada **local**, a variável **"original"** também vai ser alterada.

Array – Definição da Função

- Definição da função Maximo.

```
int Maximo(int v[]){
  int i,maior, indice;
  maior = v[0];
  indice = 0;
  for(i=0;i<N;i++){
    if(v[i] > maior){
      maior = v[i];
      indice = i;
    }
  }
  return indice;
}
```

Arrays Bidimendionais – Matrizes

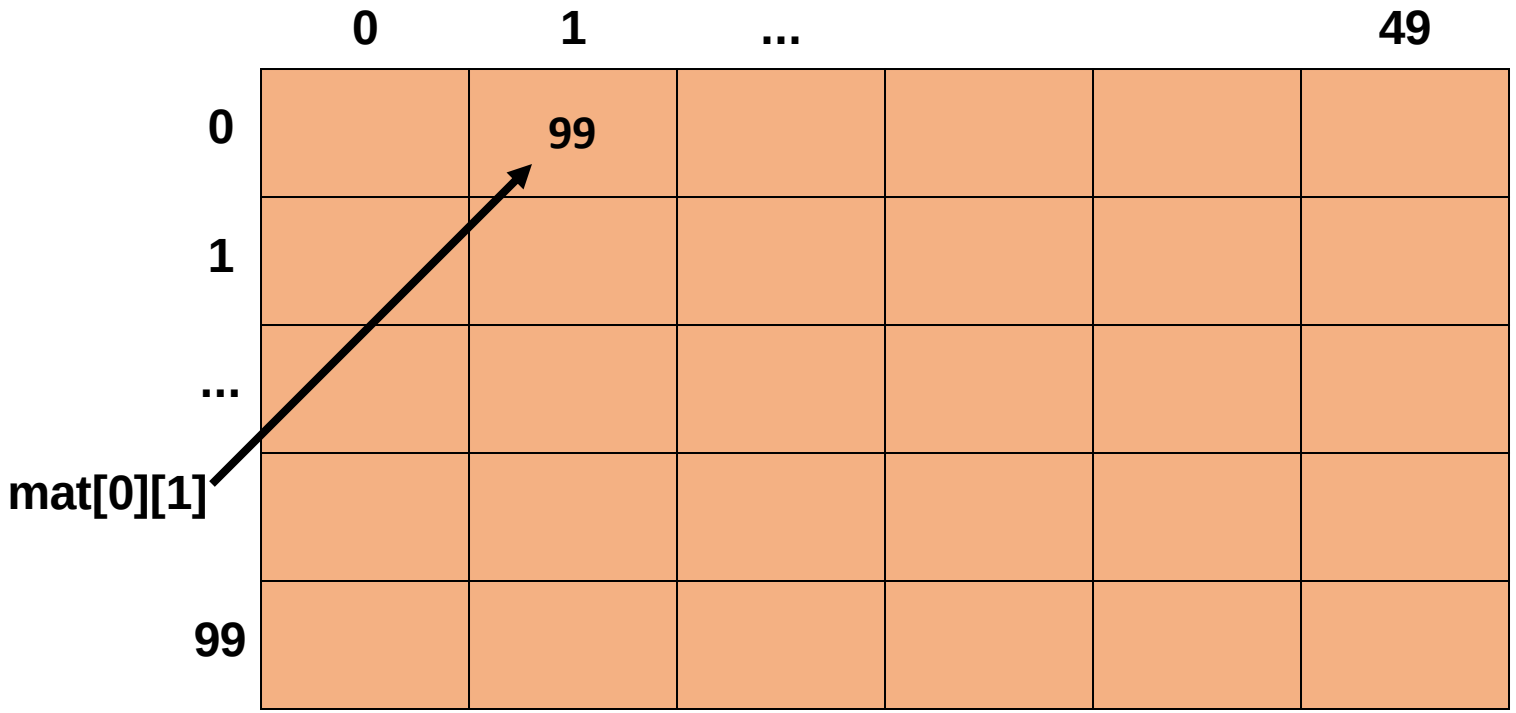
- Os arrays declarados até o momento possuem apenas uma dimensão e, portanto, são tratados como uma lista de variáveis.
  - Porém, há casos em que os dados são organizados em uma estrutura de linhas e colunas, como uma tabela. Para isso, utilizaremos um array com duas dimensões, ou seja, uma **matriz.**
  - Com isso, serão necessários dois índices para acessar uma posição: um para a linha e outro para a coluna.

```
tipo nome[linhas][colunas];
```

Arrays Bidimendionais – Matrizes

- Exemplo: Matriz com 100 linhas e 50 colunas.

```
int mat[100][50];
mat[0][1] = 99;
```



Matrizes – Leitura e Exibição

```c
#include <stdio.h>
#define N 3

int main (int argc, char *argv[]){
  int mat[N][N];
  int i, j;

  for(i=0;i<N;i++){
    for(j=0;j<N;j++){
      printf("mat[%i][%i]: ",i,j);
      scanf("%i",&mat[i][j]);
    }
  }
  printf("\n");
  for(i=0;i<N;i++){
    for(j=0;j<N;j++){
      printf("%i\t",mat[i][j]);
    }
    printf("\n");
  }
  return 0;
}
```

```
mat[0][0]: 1
mat[0][1]: 2
mat[0][2]: 3
mat[1][0]: 4
mat[1][1]: 5
mat[1][2]: 6
mat[2][0]: 7
mat[2][1]: 8
mat[2][2]: 9

1       2       3
4       5       6
7       8       9
```

Matrizes – Protótipos

- Nos protótipos, é necessário informar o tamanho da última dimensão.

```
#include <stdio.h>
#define NL 4
#define NC 3

void lerMatriz( int m[][NC]);

int main (int argc, char *argv[]){
  int mat[N];
  int max;

  lerMatriz(mat);
  return 0;
}
```

Lista de Exercícios

- Abra a pasta Downloads no VS Code
  - Utilize o comando git clone https://github.com/ECM404/lista4.git --recursive
  - Entre na pasta utilizando o comando cd lista4
  - Inicialize o diretório com o comando make install