Arrays em linguagem C

Por que usar array?

- As variáveis declaradas até agora são capazes de armazenar um único valor por vez.

```
01      #include <stdio.h>
02      #include <stdlib.h>
03      int main(){
04        float x = 10;
05        printf("x = %f\n",x);
06        x = 20;
07        printf("x = %f\n",x);
08        system("pause");
09        return 0;
10      }
```

```
Saída   x = 10.000000
        x = 20.000000
```

Array

- Array ou “Vetor” é a forma mais familiar de dados estruturados.
- Basicamente, um array é um conjunto de componentes do mesmo tipo.

Array - Problema

- Imagine o seguinte problema
  - leia as notas de uma turma de cinco estudantes e depois imprima as notas que são maiores do que a média da turma.
- Um algoritmo para esse problema poderia ser o mostrado a seguir.

Array - Solução

```
01  #include <stdio.h>
02  #include <stdlib.h>
03  int main(){
04    float n1,n2,n3,n4,n5;
05    printf("Digite a nota de 5 estudantes: ");
06    scanf("%f",&n1);
07    scanf("%f",&n2);
08    scanf("%f",&n3);
09    scanf("%f",&n4);
10    scanf("%f",&n5);
11    float media = (n1+n2+n3+n4+n5 )/5.0;
12    if(n1 > media) printf("nota: %f\n",n1);
13    if(n2 > media) printf("nota: %f\n",n2);
14    if(n3 > media) printf("nota: %f\n",n3);
15    if(n4 > media) printf("nota: %f\n",n4);
16    if(n5 > media) printf("nota: %f\n",n5);
17    system("pause");
18    return 0;
19  }
```

Array

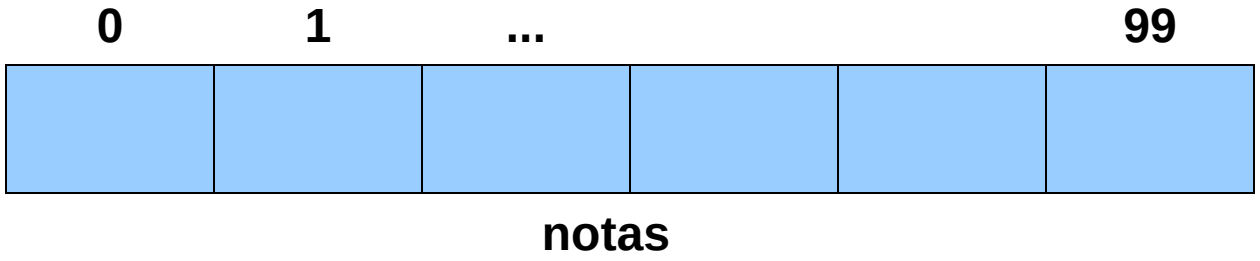
- O algoritmo anterior apresenta uma solução possível.
- Porém, essa solução é inviável para uma lista de 100 alunos.

Array

- Para 100 alunos, precisamos de:
  - Uma variável para armazenar a nota de cada aluno: 100 variáveis.
  - Um comando de leitura para cada nota: 100 scanf()
  - Um somatório de 100 notas.
  - Um comando de teste para cada aluno: 100 comandos if.
  - Um comando de impressão na tela para cada aluno: 100 printf().

Array - Definição

- As variáveis têm relação entre si
  - todas armazenam notas de alunos
- Podemos declará-las usando um ÚNICO nome para todos os 100 alunos
  - notas = conjunto de 100 números acessados por um índice = array.

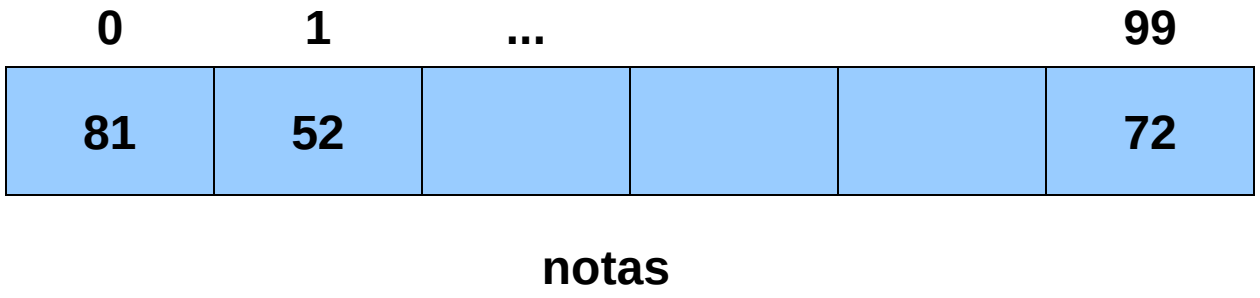


Array - Declaração

- Arrays são agrupamentos de dados adjacentes na memória. Declaração:
  - tipo_dado nome_array[tamanho];
- O comando acima define um array de nome **nome_array,** capaz de armazenar **tamanho** elementos adjacentes na memória do tipo **tipo_dado**
  - Ex: int notas[100];

Array - Definição

- Na linguagem C a numeração do array começa sempre do zero.
- Isto significa que, no exemplo anterior, os dados serão indexados de 0 a 99.
  - notas[0], notas[1], ..., notas[99]



Array - Definição

- Observação
  - Se o usuário digitar mais de 100 elementos em um array de 100 elementos, o programa tentará ler normalmente.
  - Porém, o programa os armazenará em uma parte não alocada de memória, pois o espaço alocado foi para somente 100 elementos.
  - Isto pode resultar nos mais variados erros no instante da execução do programa.

Array = varíavel

- Cada elemento do array tem todas as características de uma variável e pode aparecer em expressões e atribuições.
  - notas[2] = notas[3] + notas [20]
- Ex: somar todos os elementos de notas:

```
int soma = 0;
for(i=0;i < 100; i++)
    soma = soma + notas[i];
```

Somatório

- Ex: somando os elementos de um array de 5 elementos

```
int lista[5] = {3,51,18,2,45};
int soma = 0;
for(i=0;i < 5; i++)
        soma = soma + lista[i];
```

Variáveis		
soma	i	lista[i]
0		
3	0	3
54	1	51
72	2	18
74	3	2
119	4	45
	5	

Array - Características

- Características básicas de um Array
  - Elementos do mesmo tipo.
  - O tempo e o tipo de procedimento para acessar qualquer um dos elementos do array são iguais.
  - cada elemento componente desta estrutura tem um índice próprio segundo sua posição no conjunto

Array - Problema

- Voltando ao problema anterior
  - leia as notas de uma turma de cinco estudantes e depois imprima as notas que são maiores do que a média da turma.

Array - Solução

- Um algoritmo para esse problema usando array:

```
Para i = 1 até 5 faça
   Leia(notas[i]);
soma = 0;
Para i = 1 até 5 faça
   soma = soma + notas[i];
media = soma/5.0;
Para i = 1 até 5 faça
   Se notas[i] > media então escrever (notas[i])
```

Array - Solução

- Se ao invés de 5, fossem 100 alunos?:

```
Para i = 1 até 100 faça
   Leia(notas[i]);
soma = 0;
Para i = 1 até 100 faça
   soma = soma + notas[i];
media = soma/100.0;
Para i = 1 até 100 faça
   Se notas[i] > media então escrever (notas[i])
```

Exercício

- Exercício
  - Para um array A com 5 números inteiros, formular um algoritmo que determine o maior elemento deste array.

Exercício

```
int A[5] = {3,18,2,51,45};
int N = 5;
int Maior=A[0];
for(i=1;i<N;i++){
   if (Maior < A[i])
        Maior=A[i];
}
printf("%d", Maior);
```

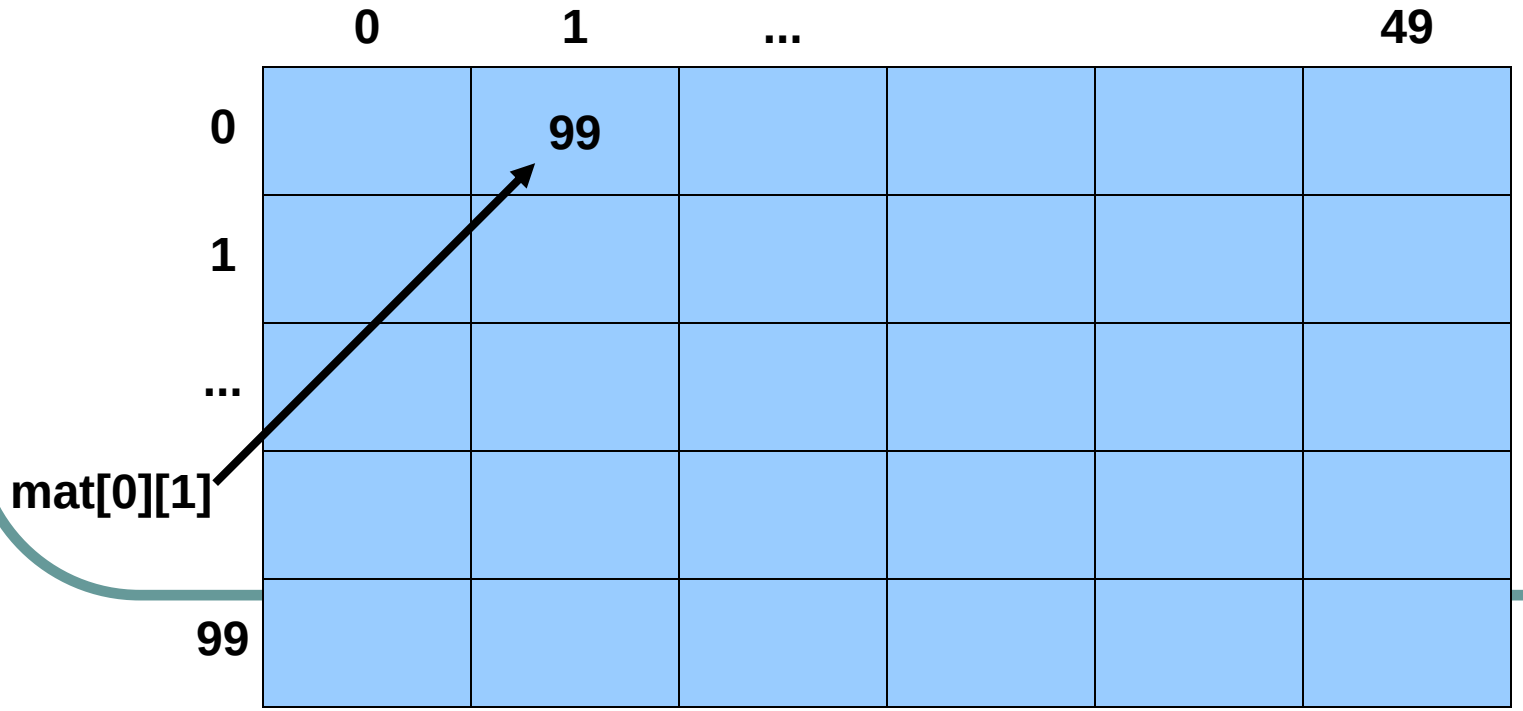
Variáveis			
Maior	i	N	A[i]
		5	
3	0		3
18	1		18
	2		2
51	3		51
	4		45
	5		

Arrays bidimensionais - matrizes

- Também chamados de “matrizes”, contém:
  - arranjados na forma de uma tabela de 2 dimensões;
  - necessita de dois índices para acessar uma posição: um para a linha e outro para a coluna
  - Índices começam sempre na posição ZERO.
- Declaração
  - tipo_variável nome_variável[linhas][colunas];

Arrays bidimensionais - matrizes

- Ex.: um array que tenha 100 linhas por 50 colunas
  - int mat[100][50];
  - mat[0][1] = 99;



Arrays bidimensionais - matrizes

- Como uma matriz possui dois índices, precisamos de dois comandos de repetição para percorrer todos os seus elementos.

Arrays bidimensionais - matrizes

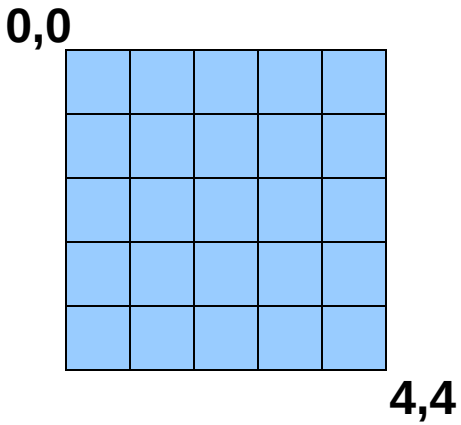
```
01  #include <stdio.h>
02  #include <stdlib.h>
03  int main(){
04    int mat[100][50];
05    int i,j;
06    for (i = 0; i < 100; i++){
07      for (j = 0; j < 50; j++){
08        printf("Digite o valor de mat[%d][%d]: ",i,j);
09        scanf("%d",&mat[i][j]);
10      }
11    }
12    system("pause");
13    return 0;
14  }
```

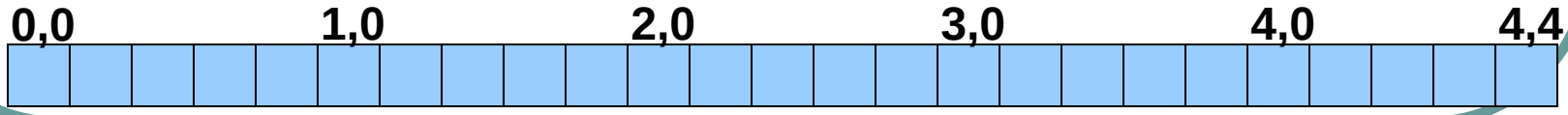
Arrays Multidimensionais

- Arrays podem ter diversas dimensões, cada uma identificada por um par de colchetes na declaração
  - int vet[5]; // 1 dimensão
  - float mat[5][5]; // 2 dimensões
  - double cub[5][5][5]; // 3 dimensões
  - int X[5][5][5][5]; // 4 dimensões

Arrays Multidimensionais

- Apesar de terem o comportamento de estruturas com mais de uma dimensão, na memória os dados são armazenados linearmente:
  - int mat[5][5];





Arrays Multidimensionais

- Um array N-dimensional funciona basicamente como outros tipos de array. Basta lembrar que o índice que varia mais rapidamente é o índice mais à direita.
  - int vet[5]; // 1 dimensão
  - float mat[5][5]; // 2 dimensões
  - double cub[5][5][5]; // 3 dimensões
  - int X[5][5][5][5]; // 4 dimensões

Exercício

- Dado um array A de 3x5 elementos inteiros, calcular a soma dos seus elementos.

Exercício

```
int soma = 0;
int i,j;
for(i=0;i<3;i++){
    for(j=0;j<5;j++){
        soma = soma + A[i][j];
    }
}
printf("%d", soma);
```

Exercício

- Dadas duas matrizes reais de dimensão 2x3, fazer um programa para calcular a soma delas.

Exercício

```
float A[2][3], B[2][3], Soma[2][3];
int i,j;
for(i=0;i<2;i++){
   for(j=0;j<3;j++){
        Soma[i][j] = A[i][j] + B[i][j];
   }
}
```

Inicialização

- Arrays podem ser inicializados com certos valores durante sua declaração. A forma geral de um array com inicialização é:
  - tipo_da_variável nome_da_variável [tam1][tam2] ... [tamN] = {lista_de_valores};

Inicialização

- A lista de valores é composta por valores (do mesmo tipo da variável) separados por vírgula. Os valores devem ser dados na ordem em que serão colocados na matriz.

```
float vect[6] = { 1.3, 4.5, 2.7, 4.1, 0.0, 100.1 };
int mat[3][4] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 };
int mat[3][4] = { {1, 2, 3, 4},{5, 6, 7, 8}, {9, 10, 11, 12}};
char str[10] = { 'J', 'o', 'a', 'o', '\0' };
char str[10] = "Joao";
char nomes[3][10] = { "Joao", "Maria", "Jose" };
```

Inicialização sem tamanho

- Inicialização sem especificação de tamanho
  - **char mess[ ] = "Linguagem C: flexibilidade e poder.";** ***//***A string mess terá tamanho 36.
  - **int matrx[ ][2] = { 1,2,2,4,3,6,4,8,5,10 };** ***//***O número de linhas de matrx será 5.

Inicialização sem tamanho

- Nesse tipo de inicialização, o compilador C vai considerar o tamanho do dado declarado como sendo o tamanho do array.
- Isto ocorre durante a compilação e não poderá mais ser mudado durante o programa.
- Isto é útil quando não queremos contar quantos caracteres serão necessários para inicializarmos uma string.

Cálculo de números primos – Crivo de Erastótenes

- Algoritmo para encontrar primos
- Mais rápido que o método que vimos anteriormente

1. Dados X números, assuma que todos são primos
2. Dado um número primo, elimine os números múltiplos deste
3. Termine até passar por toda a lista

- Imagem: http://en.wikipedia.org/wiki/Sieve_of_Eratosthenes

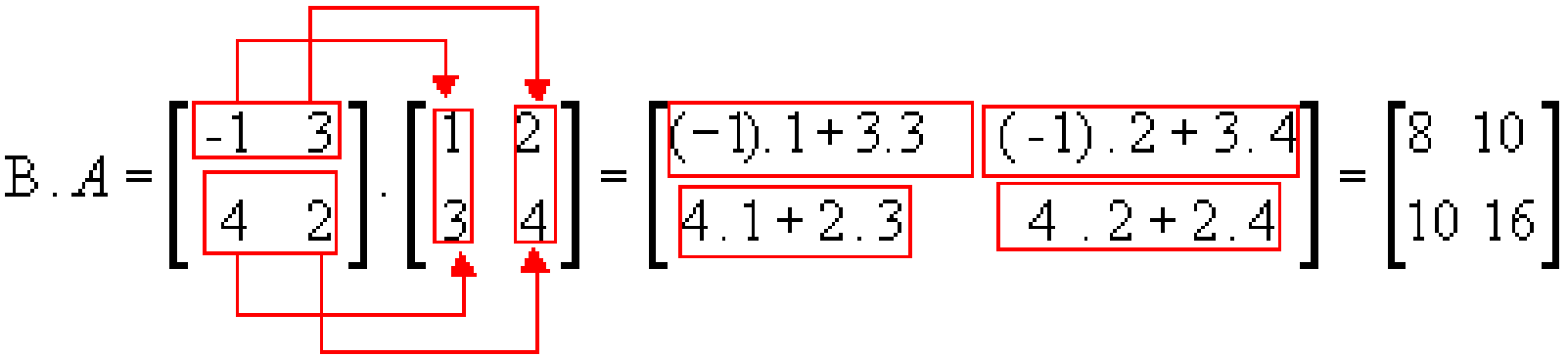
Cálculo de números primos – Crivo de Erastótenes

- Iremos criar um vetor onde o índice define o número a ser considerado.
- Vamos marcar com 1 os números primos, e com 0 os números compostos.
- Iremos ignorar a posição zero para simplificar.

```
1.  int main() {
2.     int v[101], i, primo, contaPrimo=0;
3.     for(i=1;i<101;i++) v[i] = 1;
4.     primo = 2;
5.     while(primo <= 100) {
6.        if(v[primo] == 1) {
7.             int composto = 2*primo;
8.             while(composto <= 100) {
9.                v[composto] = 0;
10.               composto += primo;
11.            }
12.       }
13.       primo++;
14.    }
15.    for(i=1;i<101;i++)
16.        if (v[i] != 0) contaPrimo++;
17.    printf("\nQuantidade de primos = %d\n",contaPrimo);
18.    return 0;
19. }
```

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

multiplicação de duas matrizes



```c
int main() {
  int linha;
  int coluna;
  int i;
  int somaprod;
  int mat1[3][3]={{1,2,3},{4,5,6},{7,8,9}};
  int mat2[3][3]={{1,0,0},{0,1,0},{0,0,1}};
  int mat3[3][3];
  int M1L=3, M1C=3, M2L=3, M2C=3;
  for(linha=0; linha<M1L; linha++)
    for(coluna=0; coluna<M2C; coluna++){
      somaprod=0;
      for(i=0; i<M1L; i++) somaprod+=mat1[linha][i]*mat2[i][coluna];
      mat3[linha][coluna]=somaprod;
    }
  //
  //imprime mat3
  //
  for(linha=0; linha<M1L; linha++){
    for(coluna=0; coluna<M2C; coluna++)
      printf("%d ", mat3[linha][coluna]);
    printf("\n");
  }
  return 0;
}
```

$$B.A = \begin{bmatrix} -1 & 3 \\ 4 & 2 \end{bmatrix} . \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} (-1).1+3.3 & (-1).2+3.4 \\ 4.1+2.3 & 4.2+2.4 \end{bmatrix} = \begin{bmatrix} 8 & 10 \\ 10 & 16 \end{bmatrix}$$