Arrays[] = {}

Problema: Dados 1000 salários. A cada taxa de ajuste inserida, informe quanto vai ser ajustado em cada salário

2300.00 3124.34 2565.00 ...
0.33
0.44
0.55

O que é array?

- Guarda um **tamanho limitado** de dados do **mesmo tipo**

- Vetor 

- “Múltiplas variáveis em um só local”

Syntax

```
TIPO NOME[TAMANHO];
```

Syntax

```
double NOME[TAMANHO];
```

Syntax

```
double salarios[TAMANHO];
```

Syntax

```
double salarios[1000];
```

Syntax

```
double salarios[1000];
```

Array de double de
1000 posições

Como é o acesso?

Acesso

```
double salarios[1000];
salarios[0]; // Primeira posição

salarios[1]; // Segunda posição

salarios[4];
```

Acesso

```
double salarios[1000];
salarios[0]; // Primeira posição
salarios[1]; // Segunda posição
salarios[4]; // Quinta posição
salarios[999];
```

Acesso

```
double salarios[1000];
salarios[0]; // Primeira posição

salarios[1]; // Segunda posição

salarios[4]; // Quinta posição

salarios[999]; //Milésima posição
```

Acesso

```
double salarios[N];
```

Como eu uso?

Leitura

```c
int idades[3];

// Idade na primeira posição
scanf("%d", &idades[0]);

// Idade na segunda posição
scanf("%d", &idades[1]);

// Idade na terceira posição
scanf("%d", &idades[2]);
```

Escrita

```c
int idades[3];

// Idade na primeira posição
printf("%d", idades[0]);

// Idade na segunda posição
printf("%d", idades[1]);

// Idade na terceira posição
printf("%d", idades[2]);
```

Operações

```
int idades[3];

idades[0] = 3;

idades[1] = idades[0] + 1;

idades[1] = idades[1] + idades[0];
idades[1] += idades[0];

idades[2] = idades[0] * 3;
```

```
int idades[100];
```

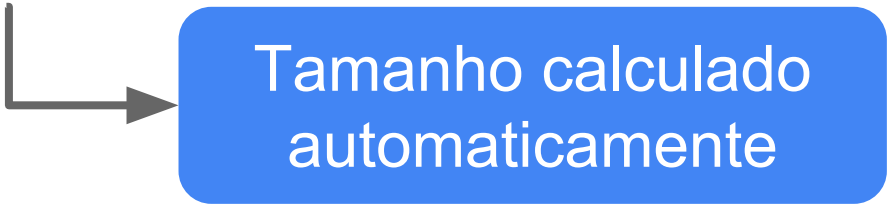
??? Leitura ???

```
int idades[100];

for(int i = 0; i < 100; ++i) {
   scanf(“%d”, &idades[i]);
}
```

Testem: Leia 10 números e imprima-os usando um array

Inicialização de um array

```
int numbers[6] = {4, 8, 15, 16, 23, 42}
int numbers[] = {4, 8, 15, 16, 23, 42}
```




```
int vetor[] = {-1, 3, -5}
```

Inicialização Parcial

```
int numbers[6] = {4, 8, 15}
```

Resto inicializado com 0


```
int vetor[10] = {}
```

Tudo inicializado com 0

E se eu quiser fazer um array de arrays?

Dica

Matriz

- Array de arrays
- Declaração, acesso, e inicialização são **similares**
- Primeiro o **número de linhas** depois o **número de colunas**

```
int m[3][4];
```

m[0][0]	m[0][1]	m[0][2]	m[0][3]
m[1][0]	m[1][1]	m[1][2]	m[1][3]
m[2][0]	m[2][1]	m[2][2]	m[2][3]

Matriz

- Array de arrays
- Declaração, acesso, e inicialização são **similares**
- Primeiro o **número de linhas** depois o **número de colunas**

Primeiro Elemento

```
int m[3][4];
```

Utiliza os indicies de 0 até N -1

m[0][0]	m[0][1]	m[0][2]	m[0][3]
m[1][0]	m[1][1]	m[1][2]	m[1][3]
m[2][0]	m[2][1]	m[2][2]	**m[2][3]**

Ultimo Elemento

Acesso

Linha Coluna

```
m[i][j];
```

```
int m[3][3];
```

4	8	15
16	23	42
108	33	13

Acesso

Linha Coluna

```
m[i][j];
```

```
m[0][0];
```

4	8	15
16	23	42
108	33	13

Acesso

Linha Coluna

```
m[i][j];
```

```
m[0][0];
```

4	8	15
16	23	42
108	33	13

Acesso

Linha Coluna

```
m[i][j];
```

```
m[1][0];
```

4	8	15
16	23	42
108	33	13

Acesso

Linha Coluna

```
m[i][j];
```

```
m[1][0];
```

4	8	15
16	23	42
108	33	13

Acesso

Linha Coluna

```
m[i][j];
```

```
m[1][2];
```

4	8	15
16	23	42
108	33	13

Acesso

Linha Coluna

```
m[i][j];
```

```
m[1][2];
```

4	8	15
16	23	**42**
108	33	13

Acesso

Linha Coluna

```
m[i][j];
```

```
m[2][1];
```

4	8	15
16	23	42
108	33	13

Acesso

Linha Coluna

```
m[i][j];
```

```
m[2][1];
```

4	8	15
16	23	42
108	**33**	13

Acesso

Linha Coluna

```
m[i][j];
```

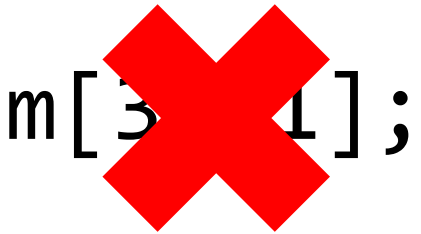
```
m[3][1];
```

4	8	15
16	23	42
108	33	13

Acesso

Linha Coluna

```
m[i][j];
```



Fora dos limites da matriz

4	8	15
16	23	42
108	33	13

Acesso

Linha Coluna

```
m[i][j];
```

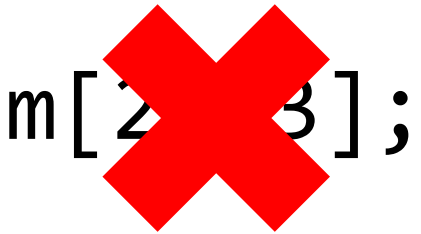
```
m[2][3];
```

4	8	15
16	23	42
108	33	13

Acesso

Linha Coluna

```
m[i][j];
```



Fora dos limites da matriz

4	8	15
16	23	42
108	33	13

Testem: Leia e imprima uma matriz 3x3

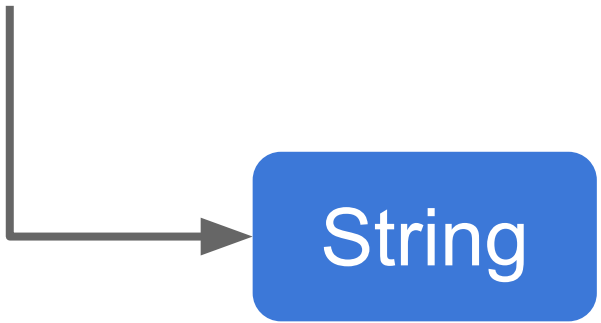
```
int m[3][3];

for(int i = 0; i < 3; ++i) {
   for(int j = 0; j < 3; ++j) {
      scanf(“%d”, &m[i][j]);
   }
}
```

Strings

String == Texto

- String é uma sequência de caracteres
- Arrays do tipo char

```
char nome[] = {‘M’, ‘A’, ‘T’, ‘H’, ‘E’, ‘U’, ‘S’};
```



Outras formas de incializar

```
char nome[] = {‘M’, ‘A’, ‘T’, ‘H’, ‘E’, ‘U’, ‘S’};
```

```
char nome[] = “MATHEUS”;
```

Aspas
Duplas

```
char nome[] = “Bazinga”;

printf(“%s\n”, nome);
```

%s formato para string

```c
char nome[] = “Bazinga”;

printf(“%s\n”, nome);

scanf(“%s”, nome);
```

%s formato para string

Serve pro printf e scanf

```c
char nome[] = “Bazinga”;

printf(“%s\n”, nome);

scanf(“%s”, nome);
```

%s formato para string

Serve pro printf e scanf

String NÃO usam o & no scanf

Testem: Leia o seu nome e imprima-o

E se fosse o nome completo?

Ruff Ghanor

```
char nome[100];

scanf("%s", nome);
printf("%s\n", nome);
```

E se fosse o nome completo?

Ruff Ghanor

```
char nome[100];

scanf("%s", nome);
printf("%s\n", nome);
```

Imprime: Ruff

E se fosse o nome completo?

Ruff Ghanor

```
char nome[100];

scanf("%s", nome);
printf("%s\n", nome);
```

Imprime: Ruff

O scanf lê tudo até o espaço

E se fosse o nome completo?

Ruff Ghanor

```
char nome[100];

scanf(“%s”, nome);
printf(“%s\n”, nome);

scanf(“%s”, nome);
printf(“%s\n”, nome);
```

E se fosse o nome completo?

Ruff Ghanor

```
char nome[100];

scanf("%s", nome);
printf("%s\n", nome);

scanf("%s", nome);
printf("%s\n", nome);
```

Imprime:

Ruff

Ghanor

Testem: Leia o nome completo, guardando ele na mesma string

```
char nome[1000];
int i = 0;
char letra = ‘0’;

while(letra != ‘\n’) {
   scanf(“%c”, &letra);
   nome[i] = letra;
   i++;
}
```

```
char nome[1000];
int i = 0;
char letra = ‘0’;

while(letra != ‘\n’) {
   scanf(“%c”, &letra);
   nome[i] = letra;
   i++;
}
```

1. No fim do loop a string nome vai conter: Maria Paula\n

2. No fim do loop a variável i vai guardar o número de letras e espaços lidos

Marcando o fim da string

Para indicar o fim da string se usa o caracter ‘\0’

```
char nome[1000];
int i = 0;
char letra = ‘0’;

while(letra != ‘\n’) {
   scanf(“%c”, &letra);
   nome[i] = letra;
   i++;
}
```

Para indicar o fim da string se usa o caracter ‘\0’

```
char nome[1000];
int i = 0;
char letra = ‘0’;

while(letra != ‘\n’) {
   scanf(“%c”, &letra);
   nome[i] = letra;
   i++;
}
nome[i] = ‘\0’;
printf(“%s”, nome);
```

Para indicar o fim da string se usa o caracter ‘\0’

Entrada:

Ana Maria

0	1	2	3	4	5	6	7	8	9
A	n	a		M	a	r	i	a	\n

i = 9

A	n	a		M	a	r	i	a	\0

```
char nome[1000];
int i = 0;
char letra = ‘0’;

while(letra != ‘\n’) {
   scanf(“%c”, &letra);
   nome[i] = letra;
   i++;
}
nome[i] = ‘\0’;
printf(“%s”, nome);
```

Coisas a se notar

Caso a string não tenha fim (‘\0’). O printf só para de imprimir quando chegar no tamanho do array

Posições com **XX** representam lixo de memória que será impresso

Imprime até aqui

	0	1	2	3	4	5	6	7	8	9	...	...	999
`printf(“%s”,`	A	n	a		M	a	r	i	a	\n	xx	xx	**xx**
`printf(“%s”,`	A	n	a		M	a	r	i	a	**\0**	xx	xx	xx

Imprime até aqui

Coisas a se notar

Quando se lê uma string com o `scanf(“%s”, nome);` o ‘\0’ é colocado automaticamente no fim da string.

`scanf(“%s”, nome);`

A	n	a	\0	xx	xx	xx	xx	xx	xx	xx	xx	xx

Porém, ele lê **até o espaço**

Lendo uma linha com o scanf

```
scanf("%[^\n]", nome);
```

Lê tudo até o \n

Lendo uma linha com o scanf

```
scanf("%[^\n]", nome);
```

Lê tudo até o \n

Entrada:

```
Ruff Ghanor\n
Feldon Druida\n
```

Lendo uma linha com o scanf

```
scanf("%[^\n]", nome);
```

Lê tudo até o \n

Entrada:

```
Ruff Ghanor\n
Feldon Druida\n
```

Já possui o \0

Lendo uma linha com o scanf

```
scanf("%[^\n]", nome);
```

Lê tudo até o \n

Não lê o \n

Entrada:

Ruff Ghanor**\n**

Feldon Druida\n

R	u	f	f		G	h	a	n	o	r	**\0**

Já possui o \0

Lendo uma linha com o scanf

```
scanf("%[^\n]", nome);

scanf("%[^\n]", nome);
```

R	u	f	f		G	h	a	n	o	r	\0

Lê tudo até o \n

O que vai ser lido na variável nome?

Entrada:

```
                \n
Feldon Druida\n
```

Lendo uma linha com o scanf

```
scanf("%[^\n]", nome);
```

R	u	f	f		G	h	a	n	o	r	\0

```
scanf("%[^\n]", nome);
```

\0

Lê tudo até o \n

O que vai ser lido na variável nome?

NADA

Entrada:

```
                \n
Feldon Druida\n
```

Lendo uma linha com o scanf

```
scanf("%[^\n]", nome);

scanf("%[^\n]", nome);
```

R	u	f	f		G	h	a	n	o	r	\0

\0

Lê tudo até o \n

Não lê o \n

Entrada:

\n

Feldon Druida\n

O que vai ser lido na variável nome?

NADA

Entre o início e o \n não há nada

Lendo uma linha com o scanf

```
scanf("%[^\n]", nome);

scanf("%[^\n]", nome);
```

R	u	f	f		G	h	a	n	o	r	\0

\0

Lê tudo até o \n

Não lê o \n

Entrada:

\n

Feldon Druida\n

Como resolve?

Lendo uma linha com o scanf

```
scanf("%[^\n]", nome);
```

R	u	f	f		G	h	a	n	o	r	\0

```
scanf(" %[^\n]", nome);
```

Espaço antes

Joga fora todos os espaços em branco: espaço, quebra de linha, tab, ...

Entrada:

```
          \n
Feldon Druida\n
```

Lendo uma linha com o scanf

```
scanf("%[^\n]", nome);
```

R	u	f	f		G	h	a	n	o	r	\0

```
scanf(" %[^\n]", nome);
```

Espaço antes

Joga fora todos os espaços em branco: espaço, quebra de linha, tab, ...

Descartado

Entrada:

\n

Feldon Druida\n

F	e	l	d	o	n		D	r	u	i	d	a	\0

Array de Strings (Matriz)

Array de Strings

```
int m[3][3];     float m[3][3];    double m[3][3];
```

Matriz de Tipos numéricos

```
char m[3][3];
```

Array de Strings

Array de Strings

```
char m[6][10];
```

Esta matriz guarda

6 strings

de no

máximo 10 caracteres

Array de Strings

Esta matriz guarda
6 strings
de no
máximo 10 caracteres

```
char m[6][10];
```

E	v	e	e	\0	xx	xx	xx	xx	xx
P	i	k	a	c	h	u	\0	xx	xx
D	i	t	t	o	\0	xx	xx	xx	xx
S	q	u	i	r	t	l	e	\0	xx
M	o	l	t	r	e	s	\0	xx	xx
M	r	.		M	i	m	e	\0	xx

Como eu imprimo o primeiro nome?

Acesso Parcial

Referencia as linhas

Linha Coluna

```
m[i][j];
```

E	v	e	e	\0	xx	xx	xx	xx	xx
P	i	k	a	c	h	u	\0	xx	xx
D	i	t	t	o	\0	xx	xx	xx	xx
S	q	u	i	r	t	l	e	\0	xx
M	o	l	t	r	e	s	\0	xx	xx
M	r	.		M	i	m	e	\0	xx

Linha Coluna

```
m[1][3];
```

E	v	e	e	\0	xx	xx	xx	xx	xx
P	i	k	a	c	h	u	\0	xx	xx
D	i	t	t	o	\0	xx	xx	xx	xx
S	q	u	i	r	t	l	e	\0	xx
M	o	l	t	r	e	s	\0	xx	xx
M	r	.		M	i	m	e	\0	xx

Linha Coluna

m[1][3];

E	v	e	e	\0	xx	xx	xx	xx	xx
P	i	k	**a**	c	h	u	\0	xx	xx
D	i	t	t	o	\0	xx	xx	xx	xx
S	q	u	i	r	t	l	e	\0	xx
M	o	l	t	r	e	s	\0	xx	xx
M	r	.		M	i	m	e	\0	xx

Acesso a Linha

Linha Coluna

```
m[1][3];
```

E	v	e	e	\0	xx	xx	xx	xx	xx
P	i	k	a	c	h	u	\0	xx	xx
D	i	t	t	o	\0	xx	xx	xx	xx
S	q	u	i	r	t	l	e	\0	xx
M	o	l	t	r	e	s	\0	xx	xx
M	r	.		M	i	m	e	\0	xx

Acesso a Linha

Linha Coluna

```
m[1][3];
```

E	v	e	e	\0	xx	xx	xx	xx	xx
P	i	k	a	c	h	u	\0	xx	xx
D	i	t	t	o	\0	xx	xx	xx	xx
S	q	u	i	r	t	l	e	\0	xx
M	o	l	t	r	e	s	\0	xx	xx
M	r	.		M	i	m	e	\0	xx

Acesso a Linha

Linha

```
m[1];
```

E	v	e	e	\0	xx	xx	xx	xx	xx
P	i	k	a	c	h	u	\0	xx	xx
D	i	t	t	o	\0	xx	xx	xx	xx
S	q	u	i	r	t	l	e	\0	xx
M	o	l	t	r	e	s	\0	xx	xx
M	r	.		M	i	m	e	\0	xx

Acesso a Linha

Linha

```
m[i];
```

Só um colchete faz, referência a linha

E	v	e	e	\0	xx	xx	xx	xx	xx
P	i	k	a	c	h	u	\0	xx	xx
D	i	t	t	o	\0	xx	xx	xx	xx
S	q	u	i	r	t	l	e	\0	xx
M	o	l	t	r	e	s	\0	xx	xx
M	r	.		M	i	m	e	\0	xx

Acesso a Linha

Só um colchete faz, referência a linha

Linha

```
m[i];
```

m[0];	E	v	e	e	\0	xx	xx	xx	xx	xx
m[1];	P	i	k	a	c	h	u	\0	xx	xx
m[2];	D	i	t	t	o	\0	xx	xx	xx	xx
m[3];	S	q	u	i	r	t	l	e	\0	xx
m[4];	M	o	l	t	r	e	s	\0	xx	xx
m[5];	M	r	.		M	i	m	e	\0	xx

Acesso a Linha

Só um colchete faz, referência a linha

Linha

```
m[i];
```

m[0];	E	v	e	e	\0	xx	xx	xx	xx	xx
m[1];	P	i	k	a	c	h	u	\0	xx	xx
m[2];	D	i	t	t	o	\0	xx	xx	xx	xx
m[3];	S	q	u	i	r	t	l	e	\0	xx
m[4];	M	o	l	t	r	e	s	\0	xx	xx
m[5];	M	r	.		M	i	m	e	\0	xx

```
printf("%s", m[0]);
```

Acesso a Linha

Só um colchete faz, referência a linha

Linha

```
m[i];
```

```
m[0];
```

E	v	e	e	**\0**	xx	xx	xx	xx	xx
P	i	k	a	c	h	u	\0	xx	xx
D	i	t	t	o	\0	xx	xx	xx	xx
S	q	u	i	r	t	l	e	\0	xx
M	o	l	t	r	e	s	\0	xx	xx
M	r	.		M	i	m	e	\0	xx

```
printf("%s", m[0]);
```

Acesso a Linha

Só um colchete faz, referência a linha

Linha

```
m[i];
```

m[0];	E	v	e	e	\0	xx	xx	xx	xx	xx
m[1];	P	i	k	a	c	h	u	\0	xx	xx
m[2];	D	i	t	t	o	\0	xx	xx	xx	xx
m[3];	S	q	u	i	r	t	l	e	\0	xx
m[4];	M	o	l	t	r	e	s	\0	xx	xx
m[5];	M	r	.		M	i	m	e	\0	xx

```
printf("%s", m[4]);
```

Acesso a Linha

Só um colchete faz, referência a linha

Linha

```
m[i];
```

	E	v	e	e	\0	xx	xx	xx	xx	xx
	P	i	k	a	c	h	u	\0	xx	xx
	D	i	t	t	o	\0	xx	xx	xx	xx
	S	q	u	i	r	t	l	e	\0	xx
m[4];	M	o	l	t	r	e	s	**\0**	xx	xx
	M	r	.		M	i	m	e	\0	xx

```
printf("%s", m[4]);
```

```
char aula[] = “FIM”;
```