



INSTITUTO FEDERAL

Norte de Minas Gerais

Campus Januária

Estruturas de Dados I

- Arrays -



Arrays

- **Imagine a solução para o seguinte problema...**
- **Sorteie 10 números inteiros aleatórios entre 0 e 20**

Soluções???



Arrays

- Imagine a solução para o seguinte problema...
- Sorteie 10 números inteiros aleatórios entre 0 e 20

não-repetitivos

Soluções???



Arrays

```
int main(){
    int num1,num2,num3,num4,num5,
        num6,num7,num8,num9,num10;

    num1 = rand() % 20;

    do{
        num2 = rand() % 20;
    }while(num2==num1);

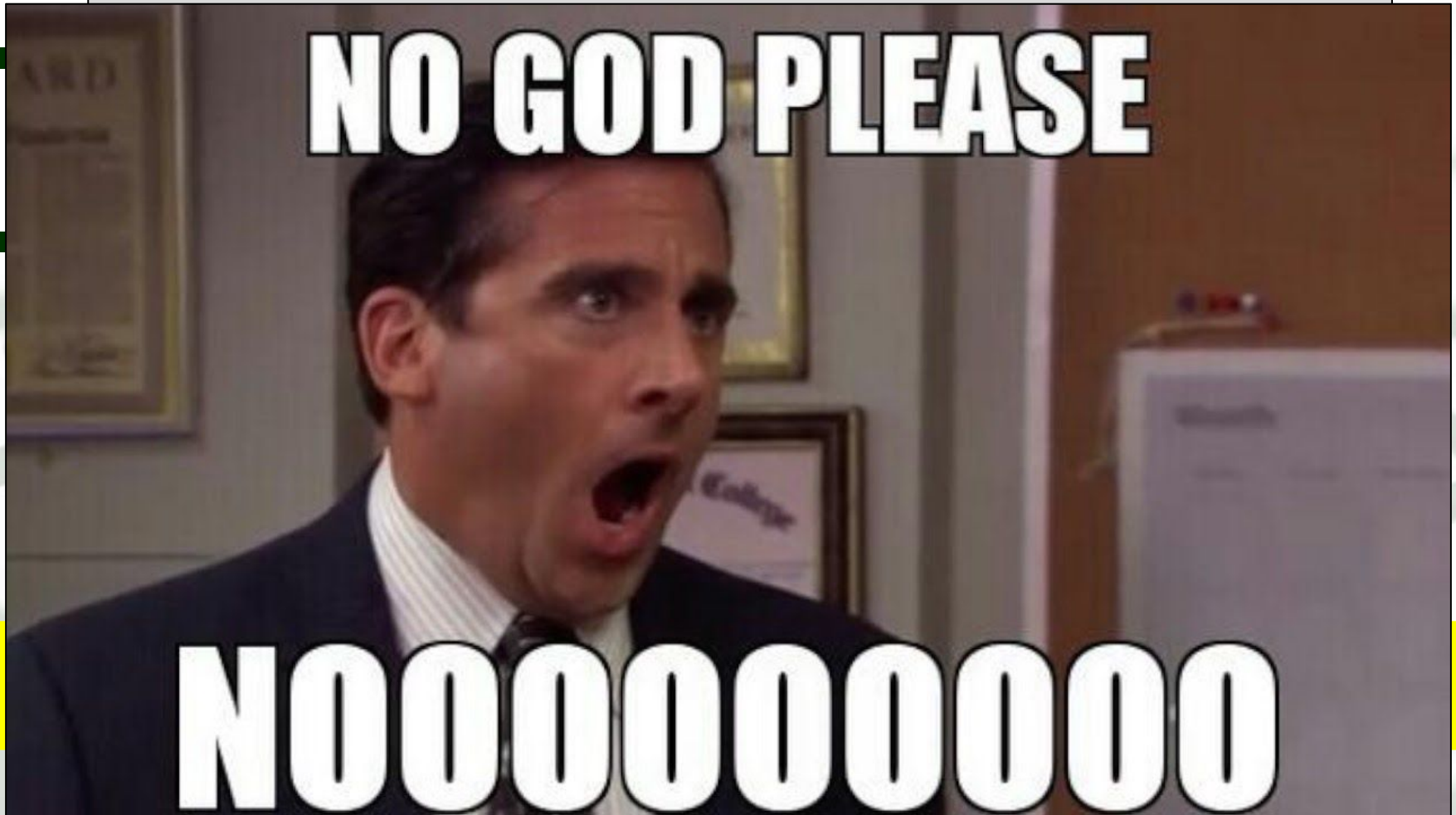
    do{
        num3 = rand() % 20;
    }while((num3==num1) || (num3==num2));

    do{
        num4 = rand() % 20;
    }while((num4==num1) || (num4==num2) || (num4==num3));
    (...)
}
```



Arrays

```
int main(){
```



```
}
```




Arrays

- **ARRAY** é uma **Estrutura de Dados** que permite a **agregação** de um conjunto de **variáveis de um mesmo tipo**, podendo este conjunto ser **referenciado** por um **identificador único**.



Arrays

- **ARRAY** é uma **Estrutura de Dados** que permite a **agregação** de um conjunto de **variáveis de um mesmo tipo**, podendo este conjunto ser **referenciado** por um **identificador único**.
- **Vetor == *Array Unidimensional***

```
int v[30];
```



Arrays

- **ARRAY** é uma **Estrutura de Dados** que permite a **agregação** de um conjunto de **variáveis de um mesmo tipo**, podendo este conjunto ser **referenciado** por um **identificador único**.

- **Vetor == *Array Unidimensional***

```
int v[30];
```

- **Matriz == *Array Multidimensional***

```
int m[10][50];
```




Representação

■ Representação de um **Vetor** (*Array Unidimensional*)

```
int main(){  
    int v[15];  
}
```



Representação

■ Representação de um **Vetor** (*Array Unidimensional*)

```
int main(){  
    int v[15];  
}
```

Índice	0	1	2	3	4	5	6	14
Valor										



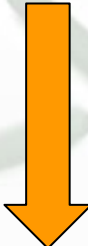
Representação

■ Representação de um **Vetor** (*Array Unidimensional*)

```
int main(){  
    int v[15];  
}
```

O primeiro índice (posição) de um array sempre será o **valor 0**.

Índice
Valor



0	1	2	3	4	5	6	14



Representação

■ Representação de um **Vetor** (*Array Unidimensional*)

```
int main(){  
    int v[15];  
}
```

O **último índice** (posição) de um **array** de tamanho N sempre será o **valor N-1**.



Índice	0	1	2	3	4	5	6	14
Valor										




Representação

■ Representação de um **Vetor** (Array Unidimensional)

```
int main(){  
    int v[15];  
}
```

*Neste exemplo, entre os índices
0 ⇔ 14 existem 15 valores inteiros.*



Índice	0	1	2	3	4	5	6	14
Valor										



Representação

■ Representação de um **Vetor** (Array Unidimensional)

```
int main(){
    int v[15];
}
```

*Ao declarar um array, é comum que o espaço alocado na memória para armazenar o conteúdo **possua resíduos (lixo de memória) de outros processos que estiveram em execução no computador.***

Índice	0	1	2	3	4	5	6	14
Valor	¶	ŧ	7	o	€	%	Û	€	¶	ث





Representação

■ Representação de um **Vetor** (Array Unidimensional)

```
int main(){
    int v[15];
    v[4] = 2;
}
```

Atribuindo valor para um índice do Array

Índice	0	1	2	3	4	5	6	14
Valor	¶	ŧ	7	๑	2	%	۞	⌘	๓	ث



Representação

■ Representação de um **Vetor** (Array Unidimensional)

```
int main(){
    int v[15];
    v[4] = 2;
    v[2] = v[4] * 4;
}
```

Atribuindo valor para um índice do Array

Índice	0	1	2	3	4	5	6	14
Valor	¶	¶	8	๑	2	%	۞	⌘	๓	ث



Representação

- É possível inicializar um Array com valores...

```
int k[6] = {0,2,4,6,8,10};
```

0	1	2	3	4	5
0	2	4	6	8	10

```
int k[10] = {0};
```

0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0



Navegando por um Array

- Para navegar (ler/escrever) nos elementos de um Array, é muito comum a utilização de uma **estrutura de repetição** (que quase sempre é o ***FOR***)

```
int main(){  
    int v[30];  
    for (int i=0; i<30; i++)  
        scanf(" %d",&v[i]);  
}
```




Navegando por um Array

```
int main(){  
    int v[10];  
    for (int i=0; i<10; i++)  
        v[i] = i*(i-1);  
}
```

`i == 0`

0	1	2	3	4	5	6	7	8	9



Navegando por um Array

```
int main(){  
    int v[10];  
    for (int i=0; i<10; i++)  
        v[i] = i*(i-1);  
}
```

$i == 0$

0	1	2	3	4	5	6	7	8	9
0									



$v[0]$



Navegando por um Array

```
int main(){  
    int v[10];  
    for (int i=0; i<10; i++)  
        v[i] = i*(i-1);  
}
```

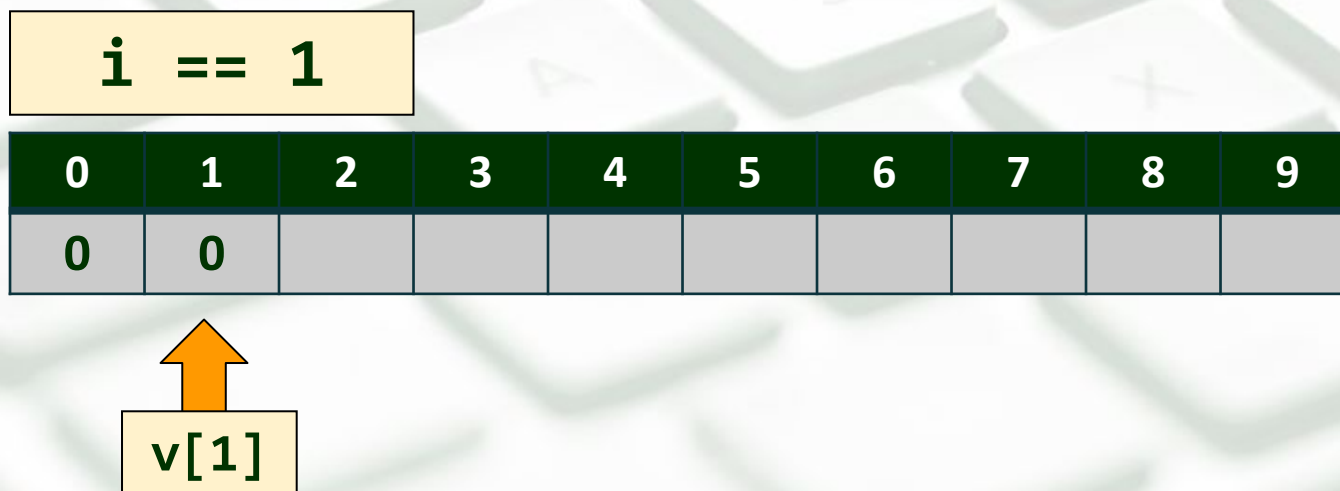
i == 1

0	1	2	3	4	5	6	7	8	9
0									



Navegando por um Array

```
int main(){  
    int v[10];  
    for (int i=0; i<10; i++)  
        v[i] = i*(i-1);  
}
```





Navegando por um Array

```
int main(){  
    int v[10];  
    for (int i=0; i<10; i++)  
        v[i] = i*(i-1);  
}
```

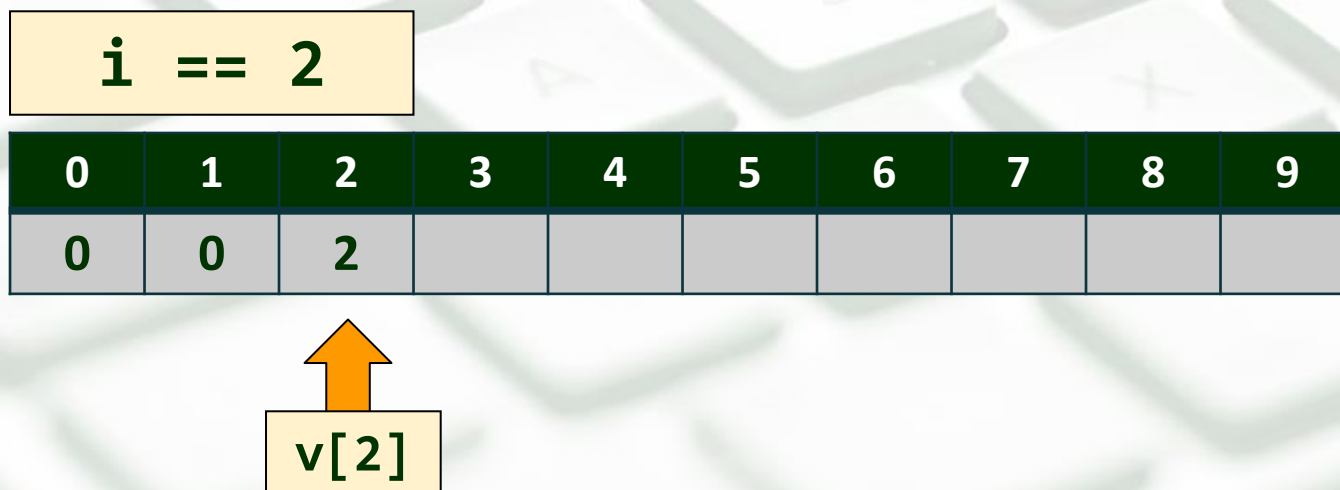
i == 2

0	1	2	3	4	5	6	7	8	9
0	0								



Navegando por um Array

```
int main(){  
    int v[10];  
    for (int i=0; i<10; i++)  
        v[i] = i*(i-1);  
}
```





Navegando por um Array

```
int main(){  
    int v[10];  
    for (int i=0; i<10; i++)  
        v[i] = i*(i-1);  
}
```

i == 3

0	1	2	3	4	5	6	7	8	9
0	0	2							



Navegando por um Array

```
int main(){  
    int v[10];  
    for (int i=0; i<10; i++)  
        v[i] = i*(i-1);  
}
```

i == 3

0	1	2	3	4	5	6	7	8	9
0	0	2	6						



v[3]



Navegando por um Array

```
int main(){  
    int v[10];  
    for (int i=0; i<10; i++)  
        v[i] = i*(i-1);  
}
```

i == 8

0	1	2	3	4	5	6	7	8	9
0	0	2	6	12	20	30	42		



Navegando por um Array

```
int main(){  
    int v[10];  
    for (int i=0; i<10; i++)  
        v[i] = i*(i-1);  
}
```

i == 8

0	1	2	3	4	5	6	7	8	9
0	0	2	6	12	20	30	42	56	



v[8]



Navegando por um Array

```
int main(){  
    int v[10];  
    for (int i=0; i<10; i++)  
        v[i] = i*(i-1);  
}
```

i == 9

0	1	2	3	4	5	6	7	8	9
0	0	2	6	12	20	30	42	56	



Navegando por um Array

```
int main(){  
    int v[10];  
    for (int i=0; i<10; i++)  
        v[i] = i*(i-1);  
}
```

i == 9

0	1	2	3	4	5	6	7	8	9
0	0	2	6	12	20	30	42	56	72



v[9]



Navegando por um Array

```
int main(){  
    int v[10];  
    for (int i=0; i<10; i++)  
        v[i] = i*(i-1);  
}
```

i == 10

0	1	2	3	4	5	6	7	8	9
0	0	2	6	12	20	30	42	56	72

Fim do Laço for



Matriz

■ Representação de uma **Matriz** (*Array 2-D*)

```
int main(){
    int m[4][10];
}
```

Índices	0	1	2	3	4	5	6	7	8	9
0		o		ŧ	@			Œ	ƞ	
1	¶	ŧ	#	o		%	Ŭ		¶	â
2			Œ	¶	Ŭ		ŧ	%	Ŭ	
3		ƞ		/	ŧ		ü			ŧ



Matriz

■ Representação de uma **Matriz** (Array 2-D)

```
int main(){
    int m[4][10];
}
```

10 Colunas

4 Linhas

Índices	0	1	2	3	4	5	6	7	8	9
0		o		ŧ	@			€	h	
1	¶	ŧ	#	o		%	ű		¶	â
2			€	¶	ű		ŧ	%	ű	
3		h		/	ŧ		ü			ŧ

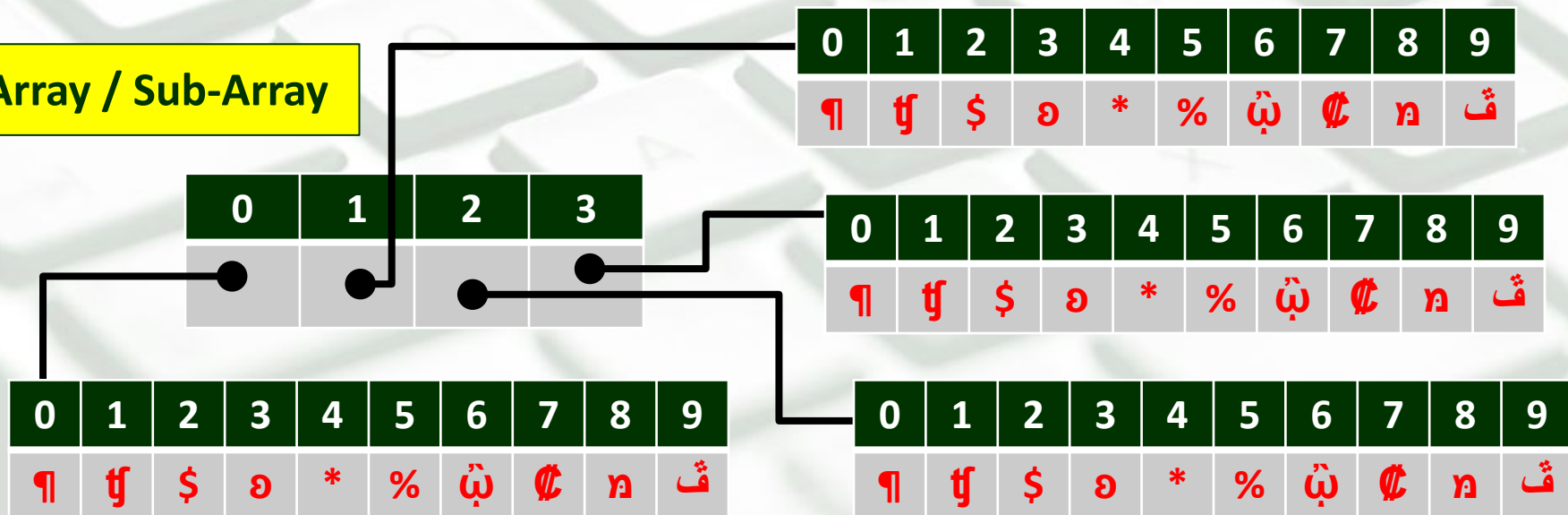


Matriz

■ Representação de uma **Matriz** (*Array 2-D*)

```
int main(){
    int m[4][10];
}
```

Array / Sub-Array





Matriz

■ Representação de uma **Matriz** (Array 2-D)

```
int main(){
    int m[4][10];
    m[3][2] = 7;
}
```

Índices	0	1	2	3	4	5	6	7	8	9
0		o		ŧ	@			Œ	ƞ	
1	¶	ŧ	#	o		%	Ŭ		¶	â
2			Œ	¶	Ŭ		ŧ	%	Ŭ	
3		ƞ	7	/	ŧ		ü			ŧ



Matriz

■ Representação de uma **Matriz** (Array 2-D)

```
int main(){
    int m[4][10];
    m[3][2] = 7;
    m[2][4] = m[3][2]*3;
}
```

Índices	0	1	2	3	4	5	6	7	8	9
0		o		f	@			¢	h	
1	¶	f	#	o		%	Û		¶	â
2			¢	¶	21		f	%	Û	
3		h	7	/	f		ü			f



Navegando por um Array

■ Array Bidimensional

```
int main(){
    int mat[30][20];
    for (int i=0; i<30; i++) //navega pelas linhas
        for (int j=0; j<20; j++) //colunas de cada linha
            scanf(" %d",&mat[i][j]);
}
```



Navegando por um Array

```
int main(){
    int mat[4][10];
    for (int i=0; i<4; i++) //navega pelas linhas
        for (int j=0; j<10; j++) //colunas de cada linha
            scanf(" %d",&mat[i][j]);
}
```

i == 0

j == 0

Índices	0	1	2	3	4	5	6	7	8	9
0	5									
1										
2										
3										



Navegando por um Array

```
int main(){
    int mat[4][10];
    for (int i=0; i<4; i++) //navega pelas linhas
        for (int j=0; j<10; j++) //colunas de cada linha
            scanf(" %d",&mat[i][j]);
}
```

i == 0

j == 1

Índices	0	1	2	3	4	5	6	7	8	9
0	5	8								
1										
2										
3										



Navegando por um Array

```
int main(){
    int mat[4][10];
    for (int i=0; i<4; i++) //navega pelas linhas
        for (int j=0; j<10; j++) //colunas de cada linha
            scanf(" %d",&mat[i][j]);
}
```

i == 0

j == 2

Índices	0	1	2	3	4	5	6	7	8	9
0	5	8	7							
1										
2										
3										



Navegando por um Array

```
int main(){
    int mat[4][10];
    for (int i=0; i<4; i++) //navega pelas linhas
        for (int j=0; j<10; j++) //colunas de cada linha
            scanf(" %d",&mat[i][j]);
}
```

i == 0

j == 3

Índices	0	1	2	3	4	5	6	7	8	9
0	5	8	7	1						
1										
2										
3										



Navegando por um Array

```
int main(){
    int mat[4][10];
    for (int i=0; i<4; i++) //navega pelas linhas
        for (int j=0; j<10; j++) //colunas de cada linha
            scanf(" %d",&mat[i][j]);
}
```

i == 0

j == 8

Índices	0	1	2	3	4	5	6	7	8	9
0	5	8	7	1	7	6	5	9	5	
1										
2										
3										



Navegando por um Array

```
int main(){
    int mat[4][10];
    for (int i=0; i<4; i++) //navega pelas linhas
        for (int j=0; j<10; j++) //colunas de cada linha
            scanf(" %d",&mat[i][j]);
}
```

i == 0

j == 9

Índices	0	1	2	3	4	5	6	7	8	9
0	5	8	7	1	7	6	5	9	5	7
1										
2										
3										



Navegando por um Array

```
int main(){
    int mat[4][10];
    for (int i=0; i<4; i++) //navega pelas linhas
        for (int j=0; j<10; j++) //colunas de cada linha
            scanf(" %d",&mat[i][j]);
}
```

i == 1

j == 0

Índices	0	1	2	3	4	5	6	7	8	9
0	5	8	7	1	7	6	5	9	5	7
1	2									
2										
3										



Navegando por um Array

```
int main(){
    int mat[4][10];
    for (int i=0; i<4; i++) //navega pelas linhas
        for (int j=0; j<10; j++) //colunas de cada linha
            scanf(" %d",&mat[i][j]);
}
```

i == 1

j == 1

Índices	0	1	2	3	4	5	6	7	8	9
0	5	8	7	1	7	6	5	9	5	7
1	2	0								
2										
3										



Navegando por um Array

```
int main(){
    int mat[4][10];
    for (int i=0; i<4; i++) //navega pelas linhas
        for (int j=0; j<10; j++) //colunas de cada linha
            scanf(" %d",&mat[i][j]);
}
```

i == 1

j == 2

Índices	0	1	2	3	4	5	6	7	8	9
0	5	8	7	1	7	6	5	9	5	7
1	2	0	3							
2										
3										



Navegando por um Array

```
int main(){
    int mat[4][10];
    for (int i=0; i<4; i++) //navega pelas linhas
        for (int j=0; j<10; j++) //colunas de cada linha
            scanf(" %d",&mat[i][j]);
}
```

i == 1

j == 3

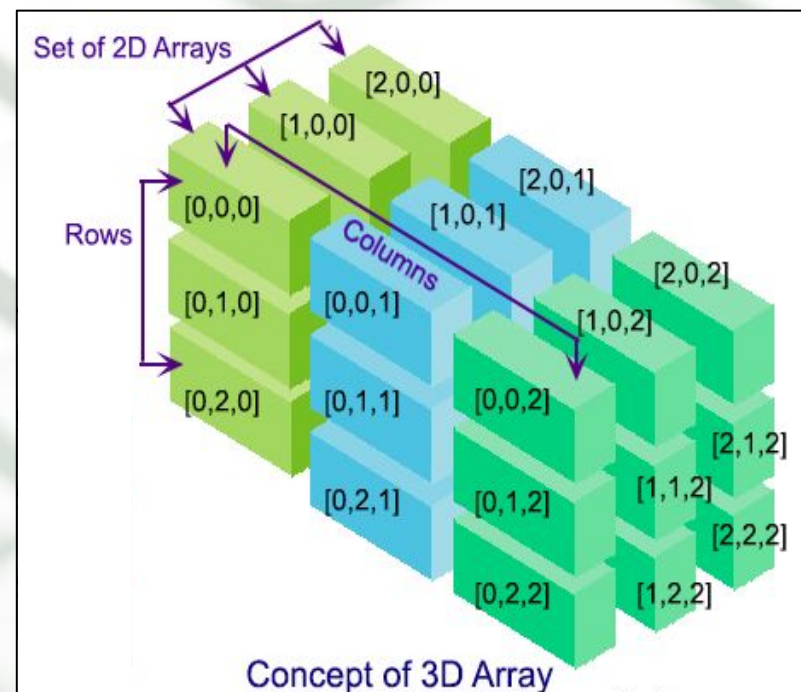
Índices	0	1	2	3	4	5	6	7	8	9
0	5	8	7	1	7	6	5	9	5	7
1	2	0	3	9						
2										
3										



Array 3D

- Representação de um *Array 3-D*
- Modelo “Cúbico”
- Aplicações muito específicas (*Data Cube*).

```
int main(){
    int cube[3][3][3];
}
```





Bora CODAR!!!



1. Para um conjunto de dados (array) de 100 valores aleatórios sorteados entre 1 e 20, faça um programa que peça ao usuário um valor X e imprima todos os números da base de dados e quantas vezes o valor X aparece nele (Marque o símbolo * ao lado de cada aparição do valor X). Repita toda operação para o mesmo conjunto de dados até $X \leq 0$.
2. Um apostador joga um dado para o ar N vezes (N definido pelo usuário). Sabendo que um dado possui 6 faces, faça um programa que simula o experimento. Como resultado, imprima quantas vezes que cada face (não quais faces) caiu para cima e o % de cada face.
3. Crie um vetor contendo N números aleatórios (sendo N múltiplo de 4). Imprima-o. Após isso, troque a metade inicial pela metade final. Imprima-o. Por fim, para cada metade inicial, troque também as sub-metades. Imprima-o.
4. Faça um programa que gera aleatoriamente uma aposta do jogo lotofácil (são 15 números não-repetitivos no intervalo entre 01 e 25). Imprima também os 10 N°s não sorteados.
5. Sorteie n°s aleatórios entre 1 e X para preencher uma tabela N x N (valor de N e X definidos pelo usuário). Imprima os n°s na tela em formato de tabela (linhas e colunas).
6. Gere uma matriz N x N onde os valores da diagonal principal sejam exatamente a soma dos demais valores existentes na sua linha e coluna.