

Estruturas de Dados e Algoritmos

Repetição, Arrays e Strings

Professores Anderson e Cesar

Estruturas de Repetição

- Permite repetir um conjunto de comandos com base em uma **condição**
 - O fluxo de execução do programa depende da **avaliação de uma ou mais expressões lógicas**
- Estruturas de Repetição em C:
 - **while**
 - **do while**
 - **for**

Estruturas de Repetição - **while**

```
#include <stdio.h>
```

```
int main() {
```

```
    int N;
```

```
    printf("Digite um valor para N\n");
```

```
    scanf("%d",&N);
```

```
    while(N != 5) {
```

```
        printf("Digite outro valor\n");
```

```
        scanf("%d",&N);
```

```
    }
```

```
    printf("Finalmente foi digitado 5\n");
```

```
    return 0;
```

```
}
```

Estruturas de Repetição - do while

```
#include <stdio.h>
```

```
int main() {  
    int N;  
    do {  
        printf("Digite outro valor\n");  
        scanf("%d",&N);  
    } while (N != 5);  
    printf("Finalmente foi digitado 5\n");  
    return 0;  
}
```

Estruturas de Repetição - **break**

```
#include<stdio.h>
```

```
int main() {  
    int N;  
    while(1) {  
        printf("Digite um valor\n");  
        scanf("%d",&N);  
        if (N == 5) break;  
    }  
    printf("Finalmente foi digitado 5\n");  
    return 0;  
}
```

Estruturas de Repetição - **continue**

```
#include<stdio.h>
```

```
int main() {  
    int n;  
    scanf("%d",&n);  
    while (n != 0) {  
        if(n < 0) continue;  
        //processa n  
        scanf("%d",&n);  
    }  
    return 0;  
}
```

Estruturas de Repetição

```
int main() {  
    int a = 5;  
    while (a != 1) {  
        if (a%2 == 0) {  
            a = a/2;  
        }  
        else {  
            a = 3*a + 1;  
        }  
    }  
    return 0;  
}
```

Repete quantas
vezes?

Estruturas de Repetição - **for**

```
#include<stdio.h>

int main() {
    int i;
    for(i = 0; i < 100; i++) {
        printf("Valor de i: %d\n", i);
    }
    return 0;
}
```

```
#include<stdio.h>

int main() {
    int i = 0;
    while(i < 100) {
        printf("Valor de i: %d\n", i);
        i++;
    }
    return 0;
}
```


Estruturas de Repetição - **for**

```
#include <stdio.h>
```

```
int main() {
```

```
    int i;
```

```
    int k;
```

```
    for(i = 0, k = 99; i < 100 && i < k; i++, k--) {
```

```
        printf("Valor de i: %d\n", i);
```

```
        printf("Valor de k: %d\n", k);
```

```
    }
```

```
    return 0;
```

```
}
```

Estruturas de Repetição - for

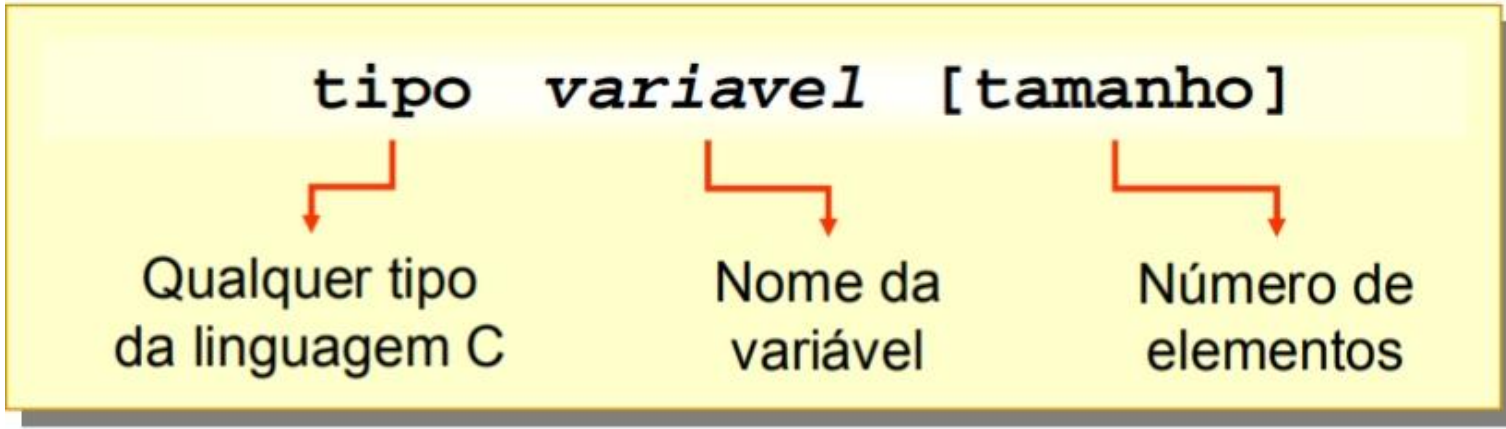
```
int main() {  
    int i, a, q, termo;  
    for(i = 5; i > 2; i--) {  
        a = i;  
        q = 3;  
        termo = a;  
        printf("%d ", (6-i));  
        while(termo <= 6*a) {  
            printf("%d ", termo);  
            termo = termo*q;  
        }  
    }  
    return 0;  
}
```

Qual a saída do programa?

Vetores

- Corresponde a uma sequência de valores do **mesmo tipo**
 - Também podem ser chamados de array (ou arranjo) ou array unidimensional
- Características:
 - Nome único para a variável
 - Tamanho fixo
 - Indexado de 0 até tamanho – 1
 - Alocados sequencialmente na memória

Declaração e Acesso a Vetores



- Declaração

```
int vetor[10];
```

```
double medidas[100];
```

- Acesso aos elementos individualmente

```
vetor[5] = 3;
```

```
vetor[0] = vetor[1] + vetor[2];
```

```
int main() {  
    float n[5], media = 5;  
    int i;  
    for(i = 0; i < 5; i++) {  
        n[i] = (i+1)/2.0;  
    }  
    return 0;  
}
```

MEMÓRIA PRINCIPAL (RAM)

Endereço físico:	13	14	15	16	17	18	19	20
	5	0.5	1.0	1.5	2.0	2.5	5.0	
Identificador da variável:	i	n[0]	n[1]	n[2]	n[3]	n[4]	med	

Inicialização de Vetores

`tipo vetor [n] = {elem0, elem1, ..., elemn-1}`

↓
Tamanho do
vetor

↓
Lista de n
valores

- Declaração de vetor com conteúdo inicial

```
int pares[5]={2,4,6,8,10};
```

- É possível omitir o tamanho do vetor

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

```
int pares[]; // Erro
```

Uso correto de vetores

Cuidado com os índices

- Errado:

```
int v[10];
```

```
...
```

```
v[-2]=5; // Efeito imprevisível
```

```
v[15]=10; // Efeito imprevisível
```

- Correto:

```
v[4]=11;
```

```
v[0]=1;
```

Uso correto de vetores

Atribuir todos os valores

- Errado:

```
int v[100];
```

```
...
```

```
v = 0;
```

- Correto:

```
int i;
```

```
for (i=0; i<100; i++) {
```

```
    v[i]=0;
```

```
}
```


Uso correto de vetores

Copiar todos os valores

- Errado:

```
int v1[100], v2[100];
```

```
...
```

```
v1 = v2;
```

- Correto:

```
int i;
```

```
for (i=0; i<100; i++) {
```

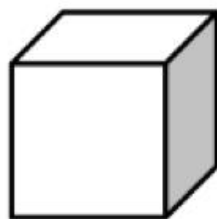
```
    v1[i]=v2[i];
```

```
}
```

Array Bidimensional

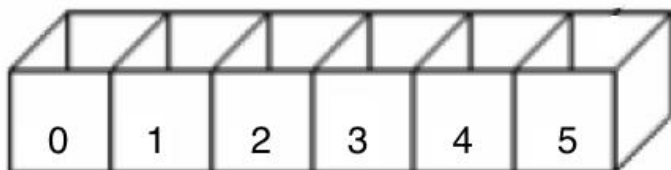
- Existem dados que são melhor modelados em forma de tabela (ou planilha)
 - Ex: planilhas excel
- Os arrays bidimensionais (ou matrizes) modelam uma tabela

Variável



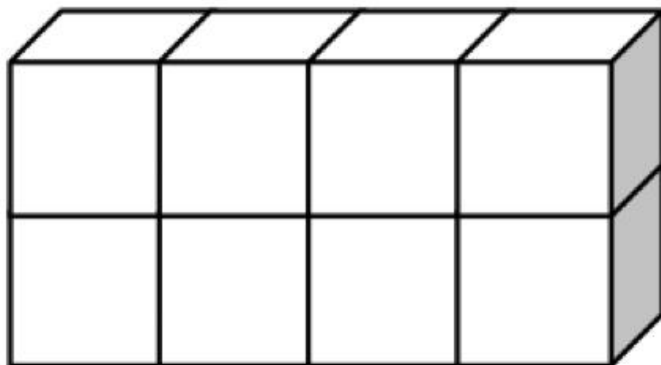
Media

Vetor



Media

Matriz



Media

Matrizes

- Declarando uma matriz

<tipo> **<identificador>**[**<qtd_linhas>**][**<qtd_colunas>**]

- Exemplos

float notas[20][4];

int imagem[800][600];

float temperatura[1000][2000];

Matrices

```
int main() {  
→ int mat[5][3]; int i,j;  
    for(i = 0; i < 5; i++) {  
        for(j = 0; j < 3; j++) {  
            mat[i][j] = i+j;  
        }  
    }  
    return 0;  
}
```

	mat		
	[0]	[1]	[2]
[0]			
[1]			
[2]			
[3]			
[4]			

Matrices

```
int main() {  
    int mat[5][3]; int i,j;  
    → for(i = 0; i < 5; i++) {  
        for(j = 0; j < 3; j++) {  
            mat[i][j] = i+j;  
        }  
    }  
    return 0;  
}
```

mat

	[0]	[1]	[2]
[0]			
[1]			
[2]			
[3]			
[4]			

i

0

Percorrendo linha 0

```
int main() {  
    int mat[5][3]; int i,j;  
    for(i = 0; i < 5; i++) {  
        for(j = 0; j < 3; j++) {  
            mat[i][j] = i+j;  
        }  
    }  
    return 0;  
}
```

mat

	[0]	[1]	[2]
[0]			
[1]			
[2]			
[3]			
[4]			

i	j
0	0

Matrizes

Percorrendo linha 0

```
int main() {  
    int mat[5][3]; int i,j;  
    for(i = 0; i < 5; i++) {  
        for(j = 0; j < 3; j++) {  
            mat[i][j] = i+j;  
        }  
    }  
    return 0;  
}
```

mat

	[0]	[1]	[2]
[0]	0		
[1]			
[2]			
[3]			
[4]			

i	j
0	0

Matrices

```
int main() {  
    int mat[5][3]; int i,j;  
    for(i = 0; i < 5; i++) {  
        for(j = 0; j < 3; j++) {  
            mat[i][j] = i+j;  
        }  
    }  
    return 0;  
}
```

mat

	[0]	[1]	[2]
[0]	0	1	
[1]			
[2]			
[3]			
[4]			

i	j
0	1

Matrizes

Percorrendo linha 0

```
int main() {  
    int mat[5][3]; int i,j;  
    for(i = 0; i < 5; i++) {  
        for(j = 0; j < 3; j++) {  
            mat[i][j] = i+j;  
        }  
    }  
    return 0;  
}
```

mat

	[0]	[1]	[2]
[0]	0	1	2
[1]			
[2]			
[3]			
[4]			

i	j
0	2

Matrizes

Percorrendo linha 1

```
int main() {  
    int mat[5][3]; int i,j;  
    for(i = 0; i < 5; i++) {  
        for(j = 0; j < 3; j++) {  
            mat[i][j] = i+j;  
        }  
    }  
    return 0;  
}
```

mat

	[0]	[1]	[2]
[0]	0	1	2
[1]			
[2]			
[3]			
[4]			

i	j
1	0

Matrizes

Como ficará a linha 1?

```
int main() {  
    int mat[5][3]; int i,j;  
    for(i = 0; i < 5; i++) {  
        for(j = 0; j < 3; j++) {  
            mat[i][j] = i+j;  
        }  
    }  
    return 0;  
}
```

mat

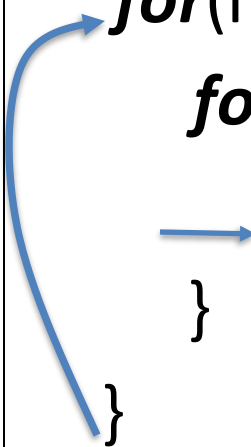
	[0]	[1]	[2]
[0]	0	1	2
[1]			
[2]			
[3]			
[4]			

i	j
1	0

Matrizes

Como ficará a linha 1?

```
int main() {  
    int mat[5][3]; int i,j;  
    for(i = 0; i < 5; i++) {  
        for(j = 0; j < 3; j++) {  
            mat[i][j] = i+j;  
        }  
    }  
    return 0;  
}
```



mat

	[0]	[1]	[2]
[0]	0	1	2
[1]	1	2	3
[2]			
[3]			
[4]			

i	j
1	2

Matrizes

Como ficará no final?

```
int main() {  
    int mat[5][3]; int i,j;  
    for(i = 0; i < 5; i++) {  
        for(j = 0; j < 3; j++) {  
            mat[i][j] = i+j;  
        }  
    }  
    return 0;  
}
```

	mat		
	[0]	[1]	[2]
[0]	0	1	2
[1]	1	2	3
[2]			
[3]			
[4]			

i	j
4	2

Matrizes

Como ficará no final?

```
int main() {  
    int mat[5][3]; int i,j;  
    for(i = 0; i < 5; i++) {  
        for(j = 0; j < 3; j++) {  
            mat[i][j] = i+j;  
        }  
    }  
    return 0;  
}
```

	mat		
	[0]	[1]	[2]
[0]	0	1	2
[1]	1	2	3
[2]	2	3	4
[3]	3	4	5
[4]	4	5	6

i	j
4	2

```
int main() {  
    int mat[4][4];  
    int soma = 0, i, j;  
    for(i = 0; i < 4; i++) {  
        for(j = 0; j < 4; j++) {  
            mat[i][j] = i*j;  
            if(i == j) {  
                soma += mat[i][j];  
            }  
        }  
    }  
    printf("%d\n", soma);  
    return 0;  
}
```

Qual a saída do algoritmo?

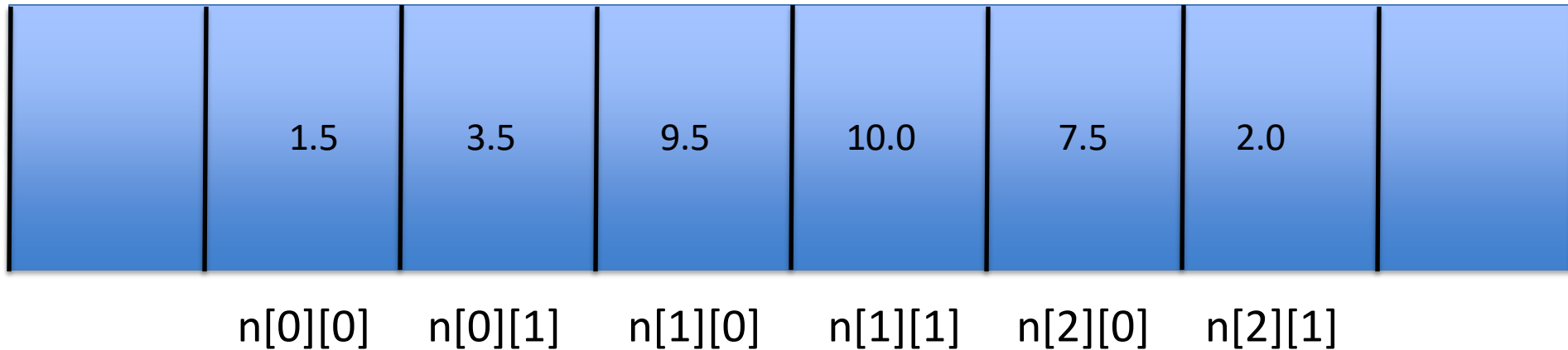
Matrizes

- E como isso fica na memória?

float notas[3][2];

São alocados 3x2 espaços contíguos de memória

MEMÓRIA PRINCIPAL (RAM)



String como Vetores

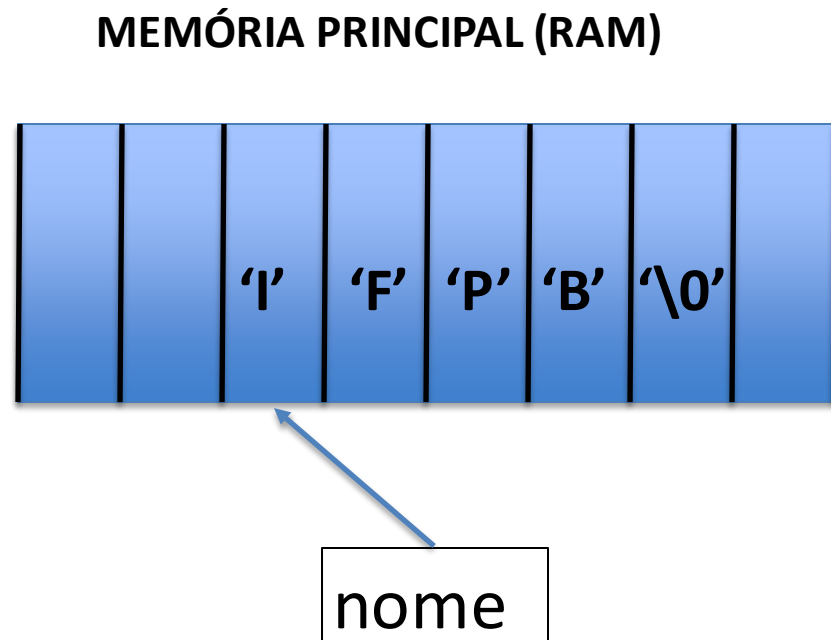
- Em C não existe o tipo **string**
 - Em C++ **string** é um objeto (não é um tipo primitivo)
 - Declaração de string como array de caracteres:

char nome[100]; //define uma string com no máximo 99 caracteres úteis

String como Vetores

- Acessando caracteres individuais de um *array* de ***char***
 - O último caractere válido sempre possui valor **'\0'**

```
int main() {  
    char nome[100]; int i;  
    scanf("%s", nome);  
    for(i = 0; nome[i] != '\0'; i++) {  
        printf("%c\n", nome[i]);  
    }  
}
```



String como Vetores

```
int main() {  
    char nome1[10], nome2[10];  
    scanf("%s %s", nome1, nome2);  
    if(nome1 == nome2) {  
        printf("São iguais.");  
    }  
    else {  
        printf("São diferentes.");  
    }  
    return 0;  
}
```

O que acontece
se o usuário
digitar "IFPB"
duas vezes?

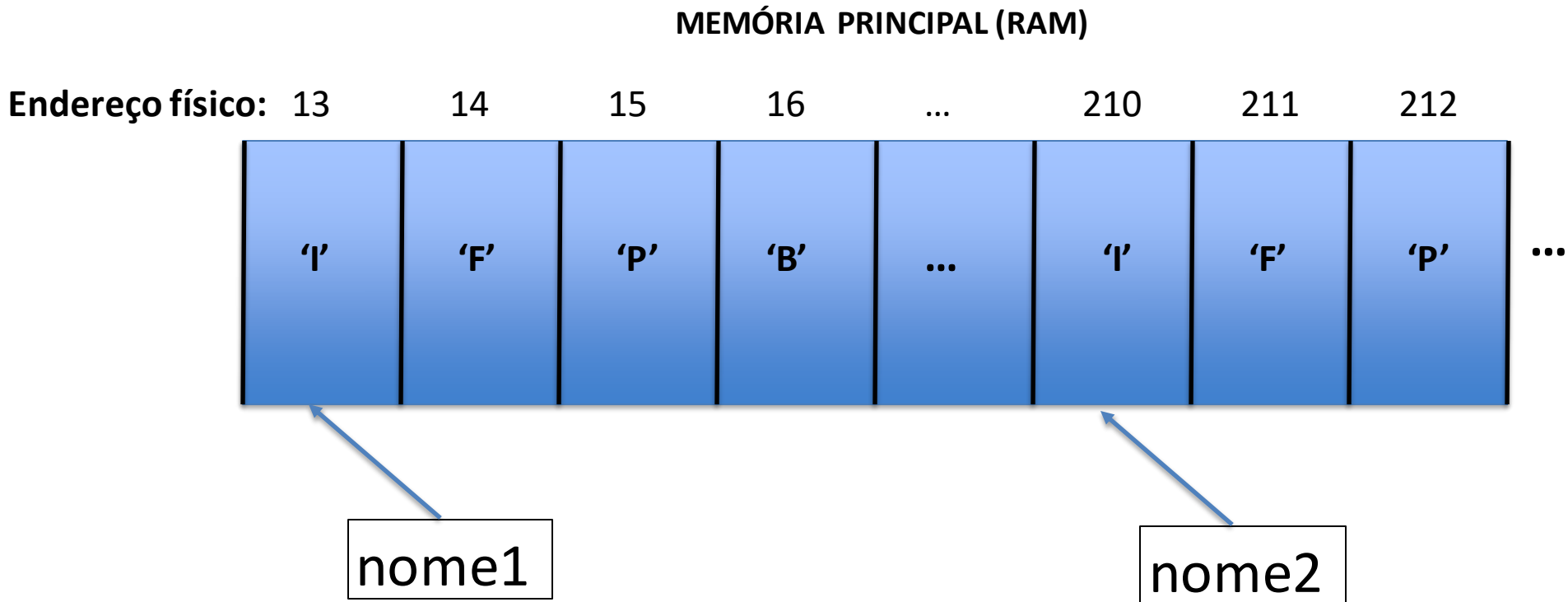
String como Vetores

```
int main() {  
    char nome1[10], nome2[10];  
    scanf("%s %s", nome1, nome2);  
    if(nome1 == nome2) {  
        printf("São iguais.");  
    }  
    else {  
        printf("São diferentes.");  
    }  
    return 0;  
}
```

Mensagem do Compilador

**warning: array
comparison
always evaluates
to false**

String como Vetores



Comparando Strings

```
#include <string.h>

int main() {
    char nome1[10], nome2[10];
    scanf("%s %s", nome1, nome2);
    if(strcmp(nome1, nome2) == 0) {
        printf("São iguais.");
    }
    else {
        printf("São diferentes.");
    }
    return 0;
}
```

Estruturas de Dados e Algoritmos

Repetição, Arrays e Strings

Professores Anderson e Cesar