

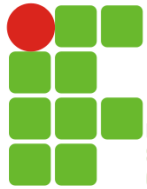


# Estrutura de Dados 1

## Revisão - Linguagem C

Antonio Angelo de Souza Tartaglia

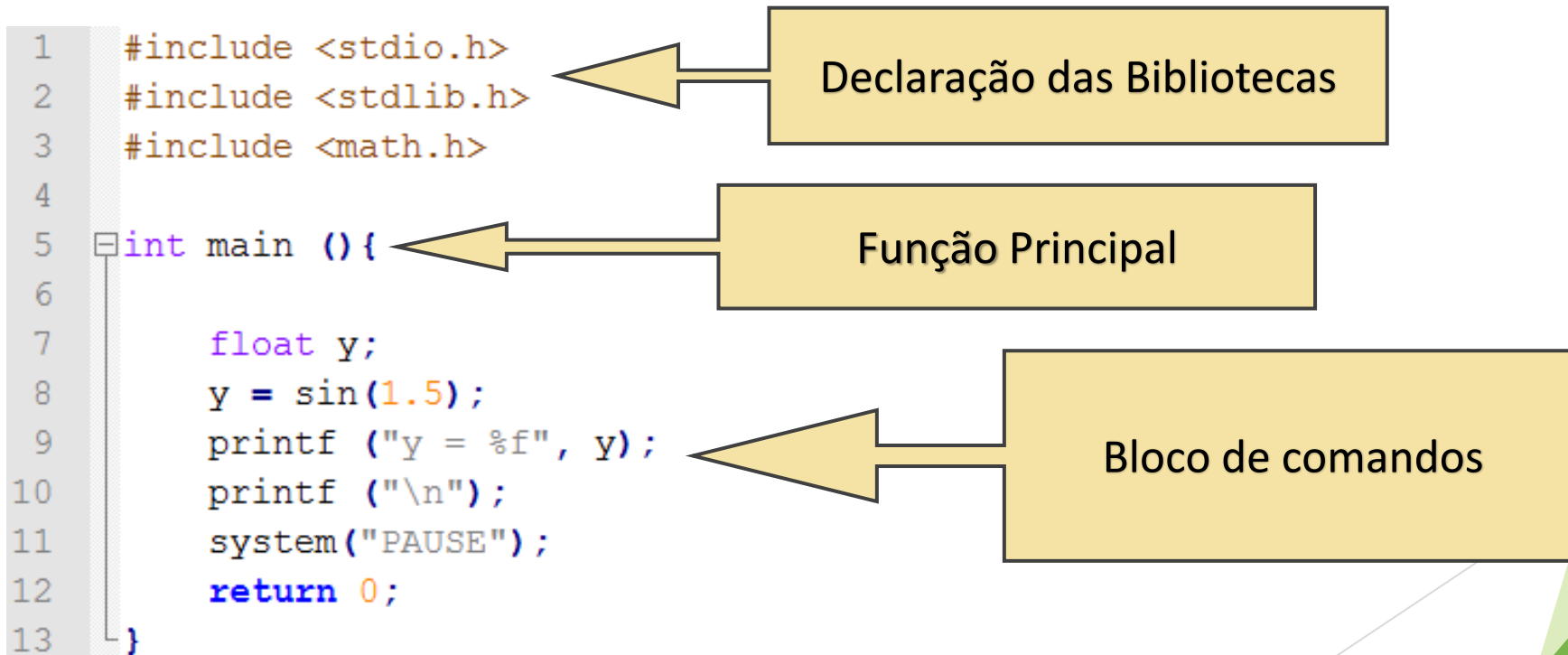
[angelot@ifsp.edu.br](mailto:angelot@ifsp.edu.br)

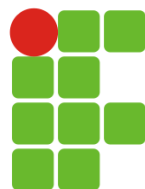


INSTITUTO FEDERAL  
SÃO PAULO  
Campus Guarulhos

# Estrutura de Dados 1 – Revisão Linguagem C

## Estrutura básica de um programa em C





INSTITUTO FEDERAL  
SÃO PAULO  
Campus Guarulhos

## Tipos de Dados

# Estrutura de Dados 1 – Revisão Linguagem C

Tipo	Tamanho em Bytes	Faixa Mínima
char	1	-127 a 127
unsigned char	1	0 a 255
signed char	1	-127 a 127
int	4	-2.147.483.648 a 2.147.483.647
unsigned int	4	0 a 4.294.967.295
signed int	4	-2.147.483.648 a 2.147.483.647
short int	2	-32.768 a 32.767
unsigned short int	2	0 a 65.535
signed short int	2	-32.768 a 32.767
long int	4	-2.147.483.648 a 2.147.483.647
signed long int	4	-2.147.483.648 a 2.147.483.647
unsigned long int	4	0 a 4.294.967.295
float	4	6 dígitos de precisão
double	8	10 dígitos de precisão
long double	10	19 dígitos de precisão

# Estrutura de Dados 1 – Revisão Linguagem C

## Códigos de formatação de argumentos – printf e scanf

Códigos printf()	Formato
%c	Caracteres simples
%d	Decimal
%e	Notação científica
%f	Ponto flutuante
%g	%e ou %f (mais curto)
%o	Octal
%s	Cadeia de caracteres
%u	Decimal sem sinal
%x	Hexadecimal
%ld	Decimal longo
%lf	Ponto flutuante longo (double)

A função **printf** (= **print formatted**) exibe na tela do monitor uma lista formatada de números, caracteres, strings, etc. O argumentos da função são strings (no caso abaixo 3 argumentos), que especificam o formato da impressão.

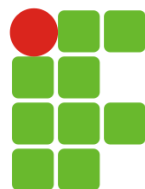
```
printf ("A média de %d e %d é %f\n", a, b, media);
```

Da mesma forma, a função **scanf** (= **scan formatted**) lê do teclado uma lista de números, caracteres, strings, etc. No exemplo abaixo, os dois primeiros argumentos da função são strings que especificam o formato da lista a ser lida. Os demais argumentos são os endereços das variáveis onde os valores lidos serão armazenados.

```
scanf ("%d %d", &a, &b);
```

# Estrutura de Dados 1 – Revisão Linguagem C

- ▶ Cada caractere em C é armazenado como um valor inteiro;
- ▶ A tabela padrão utilizada pelos computadores é a tabela ASCII.



INSTITUTO FEDERAL  
SÃO PAULO  
Campus Guarulhos

# Estrutura de Dados 1 – Revisão Linguagem C

## Tabela ASCII

Cód	Carac	Cód	Carac	Cód	Carac	Cód	Carac	Cód	Carac	Cód	Carac
32	<espaço>	33	!	34	"	35	#	36	\$	37	%
38	&	39	'	40	(	41	)	42	*	43	+
44	,	45	-	46	.	47	/	48	0	49	1
50	2	51	3	52	4	53	5	54	6	55	7
56	8	57	9	58	:	59	;	60	<	61	=
62	>	63	?	64	@	65	A	66	B	67	C
68	D	69	E	70	F	71	G	72	H	73	I
74	J	75	K	76	L	77	M	78	N	79	O
80	P	81	Q	82	R	83	S	84	T	85	U
86	V	87	W	88	X	89	Y	90	Z	91	[
92	\	93	]	94	^	95	_	96	`	97	a
98	b	99	c	100	d	101	e	102	f	103	g
104	h	105	i	106	j	107	k	108	l	109	m
110	n	111	o	112	p	113	q	114	r	115	s
116	t	117	u	118	v	119	w	120	x	121	y
122	z	123	{	124		125	}	126	~	127	<delete>

## Estrutura de Dados 1 – Revisão Linguagem C

### Caracteres de escape utilizados no comando *printf* (*strings*)

Sequência de escape	Representa
\a	Campainha (alerta)
\b	Backspace
\f	Avanço de página
\n	Nova linha
\r	Retorno de carro
\t	Tabulação horizontal
\v	Tabulação vertical
\'	Aspas simples
\ "	Aspas duplas
\\	Barra invertida
%%	Caractere Porcentagem
\?	Ponto de interrogação literal
\ooo	Caractere ASCII em notação octal
\x hh	Caractere ASCII em notação hexadecimal

# Estrutura de Dados 1 – Revisão Linguagem C

## Palavras reservadas

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while



# Estrutura de Dados 1 – Revisão Linguagem C

## Comentários em C

Comentários são uma forma de documentar os programas e torná-los mais claros para futuras consultas ou manutenção por outras pessoas:

```
/*  
  
    Este é um comentário em bloco  
  
*/  
  
/*****  
*  
*  também pode ser utilizado  *  
*      desta forma            *  
*  
*  
*****/  
  
// este é um comentário para uma linha apenas
```

# Estrutura de Dados 1 – Revisão Linguagem C

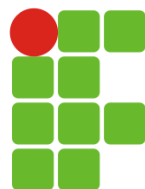
## As bibliotecas mais usadas:

<b>stdio.h</b>	<b>rotinas padrão de entrada e saída definidas pelos criadores da linguagem C.</b>
alloc.h	funções para gerenciamento de memória
float.h	funções para tratar números de ponto flutuante
<b>math.h</b>	<b>funções matemáticas</b>
stddef.h	vários tipos de dados e macro substituições
<b>stdlib.h</b>	<b>várias rotinas muito usadas, conversão, sort (ordenação), controle de memória, etc.</b>
string.h	rotinas p/ manipular strings e memória
assert.h	Macro para diagnóstico.
ctype.h	Funções para teste e conversão de caracteres (ex: isalpha(), tolower(), toupper()).
errno.h	Mnemônicas para códigos de erro.
limits.h	Constantes relacionadas com inteiros (número de bits, gama, etc.)
locale.h	Funções para informações sobre países e linguagens.
setjmp.h	Alternativa à chamada normal de funções
signal.h	Tratamento de condições excepcionais.
stdarg.h	Tratamento de funções com número e tipo de argumentos desconhecidos.
<b>string.h</b>	<b>Manipulação de cadeia de caracteres.</b>
time.h	Processamento de horas e datas.

# Estrutura de Dados 1 – Revisão Linguagem C

## Operadores Aritméticos

Operador	Descrição	Exemplo
- (unário)	Inverte o sinal de uma expressão	-10, -n, -(5 * 3 + 8 )
*	Multiplicação	3 * 5, a * b, num * 2
/	Divisão	12 / 3, num / 2
%	Módulo da divisão inteira (resto)	13 % 2, num % k
+	Adição	8 + 10, num + 2
-	Subtração	7 - 4, num - 3



INSTITUTO FEDERAL  
SÃO PAULO  
Campus Guarulhos

# Estrutura de Dados 1 – Revisão Linguagem C

## Operadores Aritméticos de atribuição

Forma Longa	Forma Resumida
$x = x + 10$	$x += 10$
$x = x - 10$	$x -= 10$
$x = x * 10$	$x *= 10$
$x = x / 10$	$x /= 10$
$x = x \% 10$	$x \% = 10$

# Estrutura de Dados 1 – Revisão Linguagem C

## Operadores Relacionais

Operador	Ação
<	Menor que
<=	Menor que ou igual
>	Maior que
>=	Maior que ou igual
==	Igual
!=	Diferente

# Estrutura de Dados 1 – Revisão Linguagem C

## Operadores Lógicos

Operador	Ação	Formato da expressão
&&	and ( <b>e</b> lógico)	A && B
	or ( <b>ou</b> lógico)	A    B
!	not ( <b>não</b> lógico)	!A

# Estrutura de Dados 1 – Revisão Linguagem C

## Comandos de controle

### ► Comandos de Seleção

- if
- else
- if-else-if
- Operador ternário
- switch

### ► Comandos de Iteração

- for
- while
- do-while

### ► Comandos de Desvio

- continue
- return
- goto
- break
- Função exit()

# Estrutura de Dados 1 – Revisão Linguagem C

## Comandos de Seleção

Seleção simples **if**

```
if (expressão){  
    //bloco de comandos  
}
```

```
#include<stdio.h>  
#include<stdlib.h>  
  
int main() {  
    int a = 20;  
    int b = 40;  
    if (a == b) {  
        printf ("a é igual a b");  
    }  
    system("pause");  
    return 0;  
}
```



# Estrutura de Dados 1 – Revisão Linguagem C

## Comandos de Seleção

Seleção composta **if-else**

```
if (expressão){  
    //bloco de comandos do if  
}else{  
    //bloco de comandos do else  
}
```

```
#include<stdio.h>  
#include<stdlib.h>  
  
int main(){  
    int a = 20;  
    int b = 40;  
  
    if (a == b){  
        printf ("a é igual a b");  
    }else{  
        printf ("a é diferente de b");  
    }  
  
    printf ("\n\n\n imprimiu");  
    system("pause");  
    return 0;  
}
```

# Estrutura de Dados 1 – Revisão Linguagem C

## Comandos de Seleção

Seleção composta **else-if**

```
if (expressão){  
    //bloco de comandos do if  
}else if(expressão){  
    //bloco de comandos do else  
}else if(expressão){  
    //bloco de comandos do else  
...  
}else if(expressão) {  
    //bloco de comandos do else  
}
```

```
#include<stdio.h>  
#include<stdlib.h>  
  
int main() {  
    int a = 30;  
    int b = 20;  
  
    if (a == b){  
        printf ("a é igual a b");  
    }else if(a > b){  
        printf ("a é maior de b");  
    }else{  
        printf ("a é menor que b");  
    }  
  
    system("pause");  
    return 0;  
}
```

```
if(expressão){  
}else{  
    if(expressão){  
    }else{  
        if(expressão){  
        }else{  
            .....  
        }  
    }  
}
```

# Estrutura de Dados 1 – Revisão Linguagem C

## Comandos de Seleção

### Operador Ternário

$(\text{expr1}) ? (\text{expr2}) : (\text{expr3})$

- Se (expr1) for (true), então (expr2) é executada;
- Caso contrário, (expr3) é executada.

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int a = 10;
    int b = 0;

    b = (a > 12) ? 100 : 200;

    printf ("%d", b);
    printf ("\n\n\n\n");
    system("pause");
    return 0;
}
```

# Estrutura de Dados 1 – Revisão Linguagem C

## Comandos de Seleção

### Seleção múltipla **switch**

- **switch** (expressão){
  - **case** expr1:
    - bloco de comandos caso 1;**break;**
  - case** expr2:
    - bloco de comandos caso 2;**break;**
  - ...
  - ...
  - default:**
    - bloco de comandos default;**break;**
- }

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

int main(){
    char ch = 'c';

    switch (ch){
        case 'a':
            printf("caracter: a");
            break;
        case 'b':
            printf("caracter:b");
            break;
        default:
            printf("Nenhuma opção");
            break;
    }

    printf ("\n\n\n\n");
    system("pause");
    return 0;
}
```

# Estrutura de Dados 1 – Revisão Linguagem C

## Comandos de iteração

### ► for

- Executa um bloco de comandos por um número definido de vezes.
- **for** (valor inicial; condição; incremento){
  - bloco de comandos
- }

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

int main(){
    int i;
    for(i = 1; i <= 10; i++){
        printf("i: %d \n", i);
    }
    printf ("\n\n\n\n");
    system("pause");
    return 0;
}
```

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

int main(){
    for(;;) {
        printf("Laço infinito!!");
    }
    printf ("\n");
    system("pause");
    return 0;
}
```

# Estrutura de Dados 1 – Revisão Linguagem C

## Comandos de iteração

### ► While

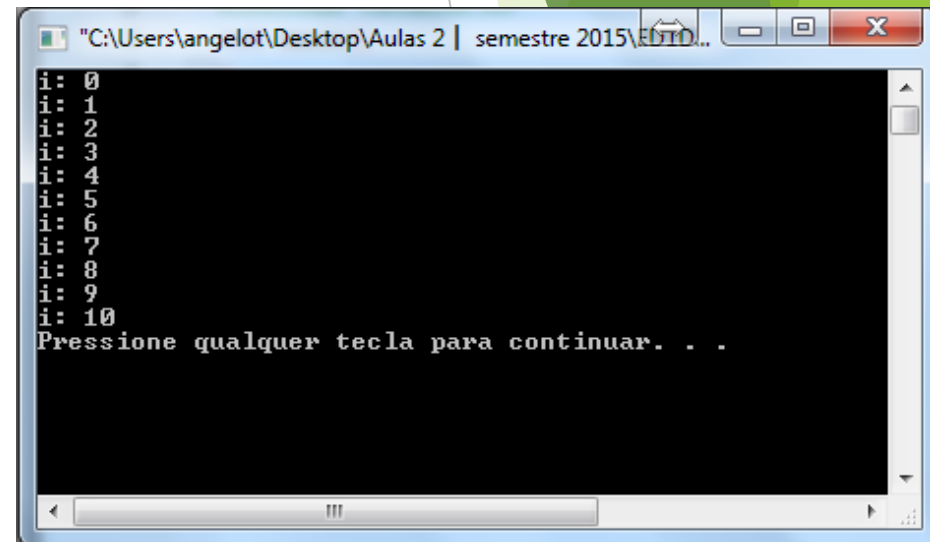
- Executa um bloco de comandos, se a condição testada for verdadeira.

- **while** (condição){
  - bloco de comandos
- }

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

int main(){
    int i = 0;
    while(i <=10){
        printf("i: %d \n", i);
        i++;
    }

    system("pause");
    return 0;
}
```



# Estrutura de Dados 1 – Revisão Linguagem C

## Comandos de iteração

### ► do-While

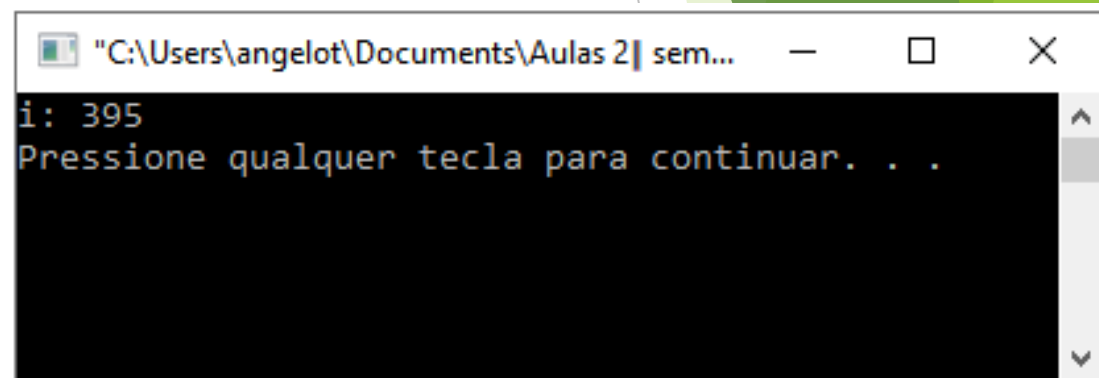
- Executa um bloco de comandos, e testa a condição no final.

- do{
  - bloco de comandos
- } while (condição);

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int i = 395;

    do{
        printf("i: %d \n", i);
        i++;
    }while(i <= 10);

    system("pause");
    return 0;
}
```



# Estrutura de Dados 1 – Revisão Linguagem C



## Comandos de desvio

### ► continue

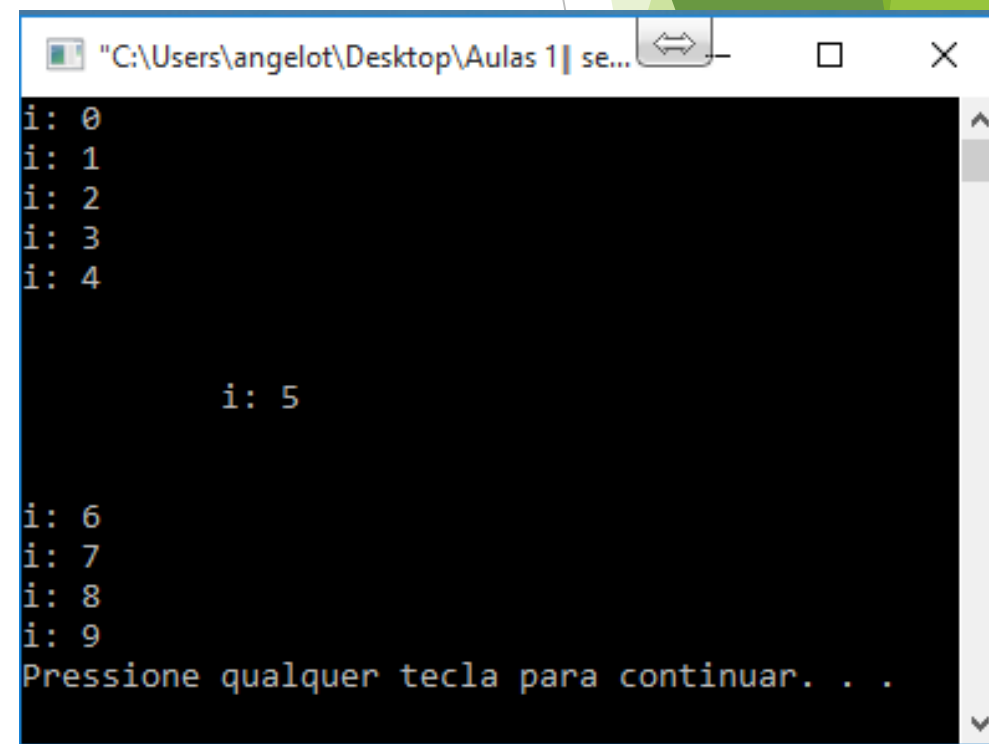
- Ignora o restante das instruções abaixo do comando **continue** dentro do laço e obriga a execução a **retornar para o início do laço**.

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

int main(){
    int i;

    for (i = 0; i < 10; i++){
        if (i == 5){
            printf("\n\n          i: %d \n\n\n", i);
            continue;
            printf("esta mensagem nao aparecerá\n");
        }
        printf ("i: %d \n", i);
    }

    system("pause");
    return 0;
}
```



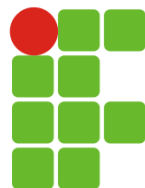
```
"C:\Users\angelot\Desktop\Aulas 1\se..."
i: 0
i: 1
i: 2
i: 3
i: 4

          i: 5

i: 6
i: 7
i: 8
i: 9
Pressione qualquer tecla para continuar. . .
```



# Estrutura de Dados 1 – Revisão Linguagem C



INSTITUTO FEDERAL  
SÃO PAULO  
Campus Guarulhos

## Comandos de desvio

### ► break

- Encerra a execução dos comandos for, while, do-while ou switch;
- O programa retorna para a primeira linha após o laço.

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

int main(){
    int i;

    for (i = 0; i < 10; i++){
        if (i == 5){
            printf("\n\n          i: %d \n\n\n", i);
            break;
            printf("esta mensagem nao aparecerá\n");
        }
        printf ("i: %d \n", i);
    }

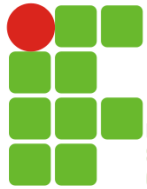
    system("pause");
    return 0;
}
```

```
"C:\Users\angelot\Desktop\Aulas 1\se..."
i: 0
i: 1
i: 2
i: 3
i: 4

          i: 5

esta mensagem nao aparecerá

Pressione qualquer tecla para continuar. . .
```



INSTITUTO FEDERAL  
SÃO PAULO  
Campus Guarulhos

# Estrutura de Dados 1 – Revisão Linguagem C

## Comandos de desvio

### ► goto

- Comando usado para saltar para outro ponto no programa.

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

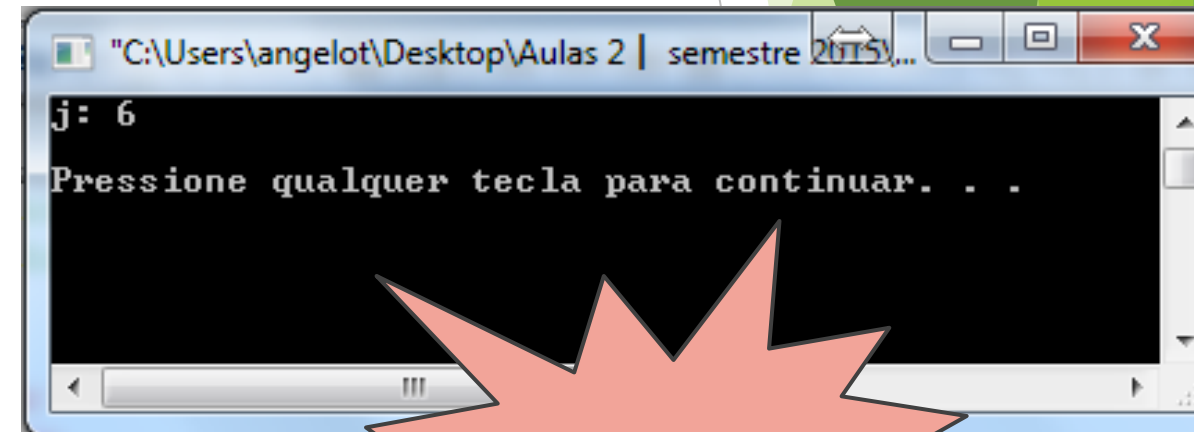
- goto rotulo;
- ...
- rotulo:
  - bloco de comandos;

```
int main() {
    int j = 5;
    if (j == 5) {
        j++;
        goto fim;
    }

    j = 10;

    fim:
    printf("j: %d\n\n", j);

    system("pause");
    return 0;
}
```



Não utilizar!!!

# Estrutura de Dados 1 – Revisão Linguagem C

## Atividade 1

- Escreva um programa em que o usuário insira um número qualquer. Se o número digitado for um da tabela abaixo, o programa deve retornar os caracteres indicados, senão, o programa deve retornar o caractere 0 (zero). Utilize o comando switch. Entregue no Moodle como atividade 1, somente o arquivo fonte (.c).

Entrada	Retorno
1	A
2	B
3	C
4	D

# Estrutura de Dados 1 – Revisão Linguagem C

## ► Atividade 2

- Reescreva o programa anterior utilizando if-else-if.

Entrada	Retorno
1	A
2	B
3	C
4	D

- Entregue no Moodle como atividade 2, somente o arquivo fonte(.c).

# Estrutura de dados 1

## Estrutura de dados

- ▶ Paralelamente ao desenvolvimento de um algoritmo, precisamos estruturar os dados que serão manipulados pelo algoritmo de tal forma que as operações mais comuns possam ser executadas eficientemente.

### Exemplo de estrutura de dados:

- Vetor ou array;
- Registros ou structs;
- Lista;
- Fila;
- Pilha;
- Árvores;
- Grafo;
- Etc.

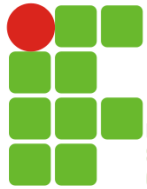
### Exemplo de uso:

- Banco de Dados;
- Compiladores;
- Planilhas
- Etc.

# Estrutura de dados 1

## Matrizes

- ▶ Estrutura de dados que podem armazenar vários valores ou dados do mesmo tipo.
  - ▶ Exemplo de tipos de dados:
    - int;
    - char;
    - float;
    - etc.
- ▶ Um vetor, ou array, é uma matriz de apenas uma dimensão.



INSTITUTO FEDERAL  
SÃO PAULO  
Campus Guarulhos

# Estrutura de dados 1

## Matrizes

- Sintaxe para declaração de uma matriz

<TIPO> nome[quantidade];

Matriz de 1 dimensão,  
ou um vetor

<TIPO> nome[qt1][qt2];

Matriz de 2 dimensões.  
qt1 = linha  
qt2 = coluna

<TIPO> nome[qt] = {valor<sub>1</sub>, valor<sub>2</sub>, ..., valor<sub>qt</sub>}

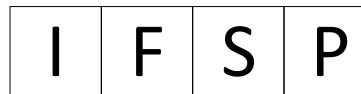
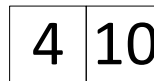
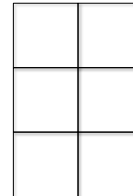
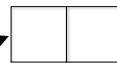
Exemplo

int notas[2];

int matriz[2][3];

int notas[2] = {4, 10};

char palavra[4] = "IFSP";



Matriz de 1 dimensão sendo  
inicializada

# Estrutura de dados 1

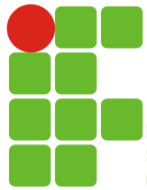
## Matrizes

### ► Representação

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

	vetor			
Valor	2	4	6	8
Índice	0	1	2	3





INSTITUTO FEDERAL  
SÃO PAULO  
Campus Guarulhos

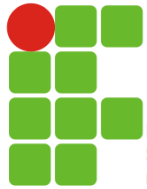
# Estrutura de dados 1

## Matrizes

- acessar um elemento na matriz:

```
int notas[4];  
notas[0] = 10;  
notas[1] = 20;  
notas[2] = 30;  
notas[3] = notas[0] + notas[2];  
printf("o resultado e; %d", notas[3]);
```

# Estrutura de dados 1



INSTITUTO FEDERAL  
SÃO PAULO  
Campus Guarulhos

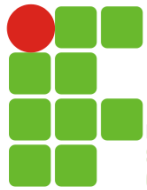
## Exemplo de Vetores - Declaração

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

int main(){
    // Declaração dos vetores com 5 posições:
    int v1[5], v2[5];
    int i = 0, j = 0;

    //preenchendo vetor 1
    for (i = 0; i < 5; i++){
        printf("\nInforme o valor do elemento %d do vetor 1: ", i + 1);
        //Leitura e inserção do valor em cada posição do vetor:
        scanf("%d", &v1[i]);
    }

    //preenchendo vetor 2
    for (j = 0; j < 5; j++){
        printf("\nInforme o valor do elemento %d do vetor 2: ", j + 1);
        //Leitura e inserção do valor em cada posição do vetor:
        scanf("%d", &v2[j]);
    }
```



INSTITUTO FEDERAL  
SÃO PAULO  
Campus Guarulhos

# Estrutura de dados 1

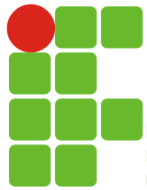
## Exemplo de Vetores - Declaração

```
for (i = 0; i < 5; i++){  
    for (j = 0; j < 5; j++){  
        //exibindo valores que são comuns aos dois vetores:  
        if(v1[i] == v2[j]){  
            printf("\nValores iguais na posição: %d e %d\n", i + 1 , j + 1);  
        }  
    }  
}  
system("pause");  
}
```

# Estrutura de dados 1

## Atividade prática 3

- ▶ Codifique o programa dos 2 vetores exemplificado anteriormente e teste seu funcionamento;
- ▶ Imprima os vetores em linhas separadas e identifique-os, em seguida exiba as posições em que possuem valores comuns ou se não possuem;
- ▶ Entregue somente o arquivo fonte (.c ) na plataforma Moodle, na entrada atividade 3.



INSTITUTO FEDERAL  
SÃO PAULO  
Campus Guarulhos

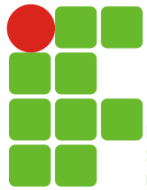
# Estrutura de dados 1

## Exemplo de Matriz 5x5

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

int main(){
    // Declaração da matriz com 5 linhas e 5 colunas:
    int mtrx[5][5], v[10];
    int i = 0, j = 0, soma = 0;

    for (i = 0; i < 5; i++){
        for (j = 0; j < 5; j++){
            printf("Digite os valores da matriz na posição: %d e %d: ", i,j);
            //efetua a leitura de todos os campos da matriz
            scanf("%d", &mtrx[i][j]);
        }
    }
}
```



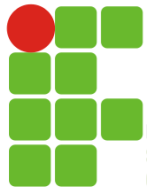
INSTITUTO FEDERAL  
SÃO PAULO  
Campus Guarulhos

# Estrutura de dados 1

## Exemplo de Matriz 5x5

- Somando as linhas da matriz

```
/*Soma as linhas da matriz, percorre por todas as colunas  
através do "j" e por todas as linhas através do "i".  
*/  
printf("\n\nTotal por linha:\n\n");  
for (i = 0; i < 5; i++){  
    for (j = 0; j < 5; j++){  
        soma = soma + mtrx[i][j];  
    }  
    printf("\nLinha %d : soma = %d\n", i + 1, soma);  
    v[i] = soma;  
    soma = 0;  
}
```



INSTITUTO FEDERAL  
SÃO PAULO  
Campus Guarulhos

# Estrutura de dados 1

## Exemplo de Matriz 5x5

- Somando as colunas da matriz

```
/*Soma as colunas da matriz, percorre todas as linhas
através do "i" e todas as colunas através do "j"
*/
soma = 0;
printf("\n\nTotal por coluna:\n\n");
for (j = 0; j < 5; j++){
    for (i = 0; i < 5; i++){
        soma = soma + mtrx[i][j];
    }
    printf("\nColuna %d : soma = %d\n", j + 1, soma);
    v[j + 5] = soma;
    soma = 0;
}
```

# Estrutura de dados 1

## Exemplo de Matriz 5x5

- Totalizando os valores calculados

```
//totalização linhas e colunas
for(i = 0; i < 5; i++){
    printf("\nOs valores da soma da linha %d são: %d\n", i + 1, v[i]);
    printf("\nOs valores da soma da coluna %d são: %d\n", i + 1 , v[i + 5]);
}
```



# Estrutura de dados 1

## Atividade prática 4

- ▶ Codifique o programa matrizes 5 x 5, em todas as suas etapas exemplificada anteriormente;
- ▶ inclua uma rotina de impressão para visualização da matriz no console (tela do terminal), que imprima em tela no formato de uma matriz, ou seja, com linhas e colunas;
- ▶ Entregue o arquivo fonte (.c ) na plataforma Moodle, na entrada atividade 4.

# Estrutura de dados 1

## Strings

- ▶ Sequência de caracteres para armazenar texto.
- ▶ Uma string é definida como uma matriz de caracteres que é terminada por um caracter NULO (“\0”);
- ▶ Exemplos:

```
char str[6] = "Texto";
```

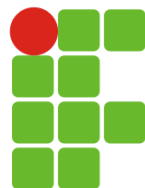
```
char str[6] = {'T','e','x','t','o','\0'};
```

```
char str[ ] = "Texto";
```

O compilador aceitará somente caracteres entre aspas simples e separados por vírgulas

Em um vetor não dimensionado, o compilador automaticamente insere “\0” ao final

# Estrutura de dados 1



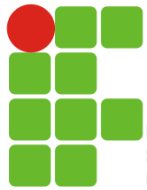
INSTITUTO FEDERAL  
SÃO PAULO  
Campus Guarulhos

## Manipulação de Strings – algumas funções

Função	Descrição
gets(s)	Lê string do dispositivo de entrada padrão e a armazena em “s”.
fgets(s,TAM, stdin)	Lê string de tamanho TAM do dispositivo de entrada padrão e armazena em s.
puts(s)	imprime string “s” no dispositivo de saída padrão.
strcpy(s1,s2)	copia s2 em s1.
strcat(s1,s2)	Concatena s2 ao final de s1.
strlen(s1)	Retorna o tamanho de s1.
strcmp(s1,s2)	Retorna 0 se (s1==s2), menor que 0 se (s1<s2) e maior que 0 se (s1>s2).
strchr(s1,ch)	Retorna um ponteiro para a primeira ocorrência de ch em s1.
strstr(s1,s2)	Retorna um ponteiro para a primeira ocorrência de s2 em s1.

A função *strcmp()* pode retornar um valor nulo (zero), positivo ou negativo. Quando as palavras comparadas são iguais, a função retorna 0. Quando as palavras comparadas são diferentes e a primeira é maior, a função retorna um valor positivo, caso contrário, a função retorna negativo, sendo que no alfabeto a “menor” letra é “a”, e a maior, “z”.

```
strcmp("bb","aa") == 1  
strcmp("aa","bb") == -1  
strcmp("bb","bb") == 0  
strcmp("aa","aa") == 0
```



INSTITUTO FEDERAL  
SÃO PAULO  
Campus Guarulhos

# Estrutura de dados 1

## Strings – Exemplo 1

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

int main(){
    char string1[20], string2[20];
    int tamString1, tamString2;
    printf("Digite duas palavras, e finalize cada uma com enter:\n");
    gets(string1);
    gets(string2);
    //obtendo o tamanho das Strings
    tamString1 = strlen(string1);
    tamString2 = strlen(string2);

    printf("Comprimentos: \nString1 = %d\nString2 = %d", tamString1, tamString2);
    //compara as Strings
    if(!strcmp(string1, string2)){
        printf("\n\nAs Strings sao iguais!");
    }else{
        printf("\n\nAs Strings nao sao iguais.");
    }
}
```

### Atenção!!

Nunca utilize a função **scanf()** para receber uma string do teclado! Utilize sempre **gets()** ou **fgets()**, que colem os caracteres até que seja pressionada a tecla enter. A função **scanf()** não aceita o caractere espaço, e quando o encontra, encerra a coleta de caracteres. Por exemplo, o nome “João da Silva”, se coletado com **scanf()**, só teria armazenado na variável de destino a string “João”.

```
Selecionar "C:\Users\angelot\Documents\Aulas\ED..."
Digite duas palavras, e finalize cada uma com enter:
carro
onibus
Comprimentos:
String1 = 5
String2 = 6

As Strings nao sao iguais.
Process returned 0 (0x0)   execution time : 7.808 s
Press any key to continue.
```

# Estrutura de dados 1



## Concatenando Strings – Exemplo 2

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

int main() {
    char string1[20], string2[20];
    printf("Digite duas palavras, e finalize cada uma com enter:\n");
    gets(string1);      O N I B U S \0
    gets(string2);      C A R R O \0
    //Concatenando Strings
    strcat(string1, string2);
    printf("\nString concatenada: %s", string1);
    printf("\n\n\n");
}
```

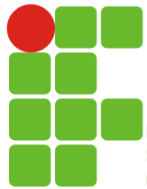
O N I B U S C A R R O \0

```
"C:\Users\angelot\Documents\Aulas\ED1D2\Aulas\..."
Digite duas palavras, e finalize cada uma com enter:
onibus
carro

String concatenada: onibuscarro

Process returned 0 (0x0)   execution time : 4.107 s
Press any key to continue.
```

# Estrutura de dados 1



INSTITUTO FEDERAL  
SÃO PAULO  
Campus Guarulhos

## Strings – Exemplo 1

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

int main(){
    char string1[20] = "Brasileiro";
    char string2[20] = "Brasil";
    char letra = 's';
    //Comparando Strings
    if(strchr(string1, "s")){//strchr devolve valor diferente de 0
        printf("\nO caracter \"%c\" esta na string \"%s\"", letra, string1);
    }
    if(strstr(string1, string2)){//strstr devolve valor diferente de 0
        printf("\n\nA String \"%s\" esta na string \"%s\"", string2, string1);
    }

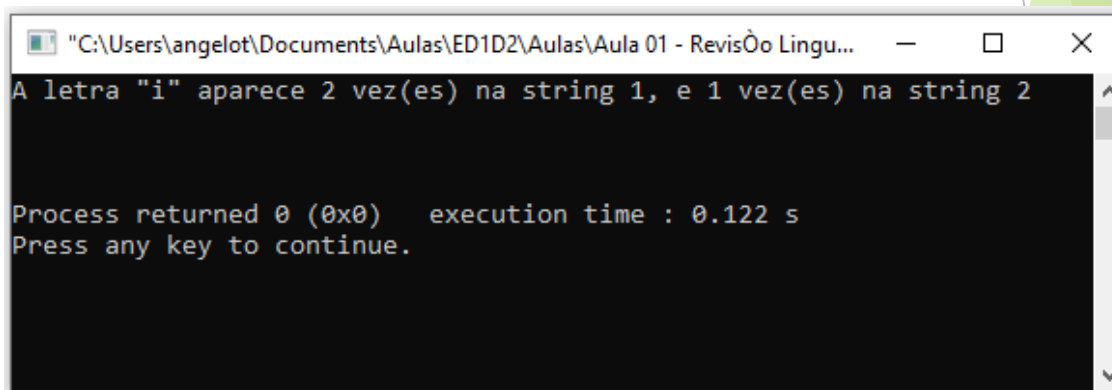
    printf("\n\n\n");
}
```

# Estrutura de dados 1

## Strings – Exemplo 2

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

int main(){
    char string1[20] = "Brasileiro";
    char string2[20] = "Brasil";
    char letra = 'i';
    int x, cont1 = 0, cont2 = 0;
    //Contando a ocorrencia de "letra" nas strings
    for(x = 0; x < strlen(string1); x++){
        if(string1[x] == letra){
            cont1++;
        }
    }
    for(x = 0; x < strlen(string2); x++){
        if(string2[x] == letra){
            cont2++;
        }
    }
    printf("A letra \"%c\" aparece %d vez(es) na string 1,|", letra, cont1);
    printf(" e %d vez(es) na string 2", cont2);
    printf("\n\n\n");
}
```



```
"C:\Users\angelot\Documents\Aulas\ED1D2\Aulas\Aula 01 - Revisão Lingu..."
A letra "i" aparece 2 vez(es) na string 1, e 1 vez(es) na string 2

Process returned 0 (0x0)   execution time : 0.122 s
Press any key to continue.
```

# Estrutura de dados 1

## Atividade prática 5

- ▶ Digite o exemplo anterior e verifique seu funcionamento;
- ▶ Altere-o para que o usuário insira via teclado as duas strings e o caracter a ser procurado nas buscas dentro das 2 strings informadas. Inclua também nessa busca dentro das strings a possibilidade do usuário fornecer parte de uma palavra (como exemplo, você pode pensar em uma busca de nome parcial);
- ▶ Entregue o arquivo fonte (.c ) na plataforma Moodle, na entrada atividade 5.



# Estrutura de dados 1

## Atividade prática 6

- ▶ Faça um programa que receba uma string do usuário (um pequeno texto), e conte quantos espaços em branco o texto possui e exiba o resultado.
- ▶ Em seguida, elimine estes espaços em branco e imprima a string resultante.