



Estrutura de Dados 1

Linguagem C - Revisão

Antonio Angelo de Souza Tartaglia
angelot@ifsp.edu.br

Estruturas de dados 1



Estrutura básica de um programa em linguagem C

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

Área de Declaração das
Bibliotecas

Área de declarações globais. Tudo o que for
colocado aqui, estará disponível a todo o
programa

Função Principal

```
int main (int argc, char *argv[]) {
    float y;
    y = sin(1.5);
    printf ("y = %f", y);
    printf ("\n");
    system("PAUSE");
    return 0;
}
```

Bloco de
comandos

Escopo: Tudo que está
dentro das { }

Escopo: Tudo que
está dentro das { }

Estruturas de dados 1



- Tipos de dados e seus tamanhos

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

Estruturas de dados 1



- Códigos de formatação para argumentos variáveis das funções – `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 4 argumentos), em que o primeiro argumento especifica quantos são no total e o formato a ser utilizado em sua exibição.

```
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, o primeiro argumento da função é uma *string* que especifica qual o formato dos dados a serem lidos e quantos serão. Os demais argumentos são os **endereços** das variáveis (passados pelo "&"), onde os valores que serão lidos vão ser armazenados.

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

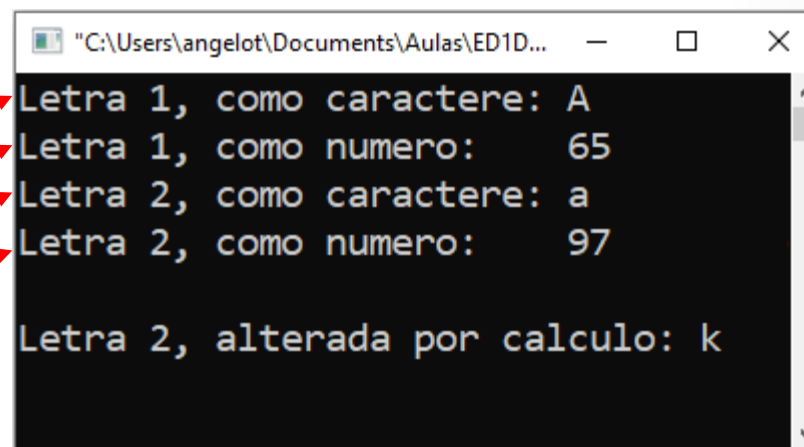
Estruturas de dados 1



- Cada caractere em C é armazenado com um valor numérico inteiro:

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

```
int main () {
    char letra_1 = 'A';
    char letra_2 = 'a';
    printf("Letra 1, como caractere: %c \n", letra_1);
    printf("Letra 1, como numero: %d \n", letra_1);
    printf("Letra 2, como caractere: %c \n", letra_2);
    printf("Letra 2, como numero: %d \n", letra_2);
    printf("\n");
    printf("Letra 2, alterada por calculo: %c \n", letra_2 + 10);
    return 0;
}
```



```
"C:\Users\angelot\Documents\Aulas\ED1D..."
Letra 1, como caractere: A
Letra 1, como numero: 65
Letra 2, como caractere: a
Letra 2, como numero: 97

Letra 2, alterada por calculo: k
```

- Para a representação como caracteres é utilizada a tabela padrão de conversão, a tabela ASCII.

Estruturas de dados 1



- 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>

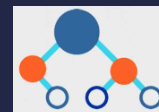
Estruturas de dados 1



Caracteres de escape utilizados (para execução ou exibição) no comando `printf()`

Sequência de escape	Representa
<code>\a</code>	Campainha (alerta)
<code>\b</code>	Backspace
<code>\f</code>	Avanço de página
<code>\n</code>	Nova linha (o mesmo que a tecla enter)
<code>\r</code>	Retorno de carro
<code>\t</code>	Tabulação horizontal
<code>\v</code>	Tabulação vertical
<code>\'</code>	Aspas simples
<code>\"</code>	Aspas duplas
<code>\\</code>	Barra invertida
<code>%%</code>	Caractere Porcentagem
<code>\?</code>	Ponto de interrogação literal
<code>\ooo</code>	Caractere ASCII em notação octal
<code>\xhh</code>	Caractere ASCII em notação hexadecimal

Estruturas de dados 1

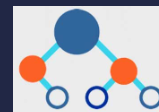


Palavras reservadas (comandos da linguagem)

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

- Não utilizar nenhuma dessas palavras para nomear variáveis e funções!!

Estruturas de dados 1



- Comentários em C
 - Comentários são uma das formas de documentar os seus programas e torna-los mais claros para futuras consultas ou manutenção por outros programadores. Sua utilização não altera o funcionamento do seu programa. Quando encontrados pelo compilador, no momento da compilação, simplesmente são ignorados e descartados no programa final, já compilado.

```
/*
```

Este é um comentário em bloco

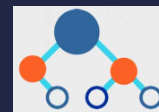
```
*/
```

```
/******  
*  
*  também pode ser utilizado  *  
*      desta forma            *  
*  
******/
```

Também chamados
“DocStrings”

```
// este é um comentário para uma linha apenas
```

Estruturas de dados 1



- As Bibliotecas mais utilizadas:

- stdio.h** rotinas padrão de entrada e saída de dados definidas pelos criadores da linguagem C;
- alloc.h** funções para gerenciamento de memória;
- float.h** funções para tratar números de ponto flutuante;
- math.h** funções matemáticas;
- stddef.h** definições de vários tipos de dados e macro para substituições;
- stdlib.h** várias rotinas muito usadas, conversão de tipos, *sort* (ordenação), controle de memória, etc;
- string.h** rotinas p/ manipular *strings* (cadeias de caracteres), e memória;
- assert.h** Macro para diagnóstico;
- ctype.h** Funções para teste e conversão de caracteres (ex: *isalpha()*, *tolower()*, *toupper()*);
- errno.h** Mnemônicos para códigos de erro;
- limits.h** Constantes relacionadas com inteiros (valores máximos, número de bits, gama, etc.);
- locale.h** def. aspectos relacionados à formatação de números, datas, moeda e texto de uma localidade;
- 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, ex.: *printf* e *scanf*;
- time.h** Processamento de horas e datas.



- Operadores aritmético

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	$7 - 4$, $num - 3$

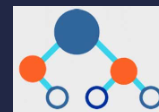
- Unário porque atua em somente um operando.



- Operadores de atribuição

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

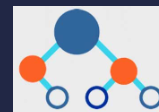
Estruturas de dados 1



- Operadores relacionais

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

Estruturas de dados 1



- Operadores lógicos

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

Estruturas de dados 1



- Comandos de controle

Seleção

- `if`
- `Else`
- `if - else if`
- `? - operador Ternário`
- `switch`

Comandos de desvio

- `continue`
- `return`
- `goto`
- `break`
- a função `exit()`

Comandos de iteração

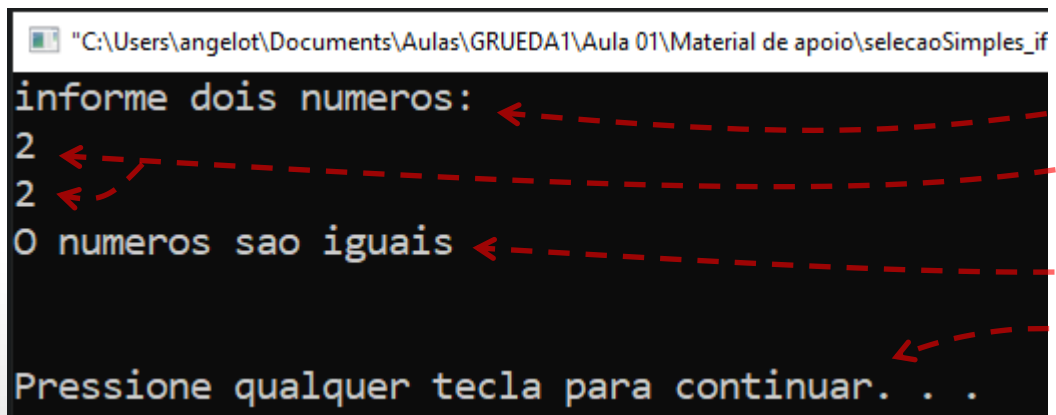
- `for`
- `while`
- `do-while`

Estruturas de dados 1

Comandos de seleção

Seleção simples if

```
if('expressão avaliada'){  
    /*  
    bloco de comandos à  
    executar, caso expressão  
    seja verdadeira  
    */  
}
```



```
"C:\Users\angelot\Documents\Aulas\GRUEDA1\Aula 01\Material de apoio\selecaoSimples_if  
informe dois numeros: <  
2 <  
2 <  
0 numeros sao iguais <  
Pressione qualquer tecla para continuar. . .
```

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

```
int main(){  
    int a, b;  
    printf("informe dois numeros: \n");  
    scanf(" %d %d", &a, &b);  
  
    if(a == b){  
        printf("Os numeros sao iguais \n\n\n");  
    }  
    system("pause");  
    return 0;  
}
```



Estruturas de dados 1

Comandos de seleção

Seleção composta if - else

```
if('expressão avaliada'){  
    /*  
    bloco de comandos à executar,  
    caso expressão seja verdadeira  
    */  
}else{  
    /*  
    bloco de comandos à executar,  
    caso expressão seja falsa  
    */  
}
```

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

```
int main(){  
    int a, b;  
    printf("informe dois numeros: \n");  
    scanf(" %d %d", &a, &b);  
  
    if(a == b){  
        printf("Os numeros sao iguais \n\n\n");  
    }else{  
        printf("os numeros sao diferentes \n\n\n");  
    }  
    system("pause");  
    return 0;  
}
```

"C:\Users\angelot\Documents\Aulas\GRUEDA1\Aula 01\Material de apoio\selecaoComp

```
informe dois numeros:  
1  
2  
os numeros sao diferentes  
  
Pressione qualquer tecla para continuar. . .
```



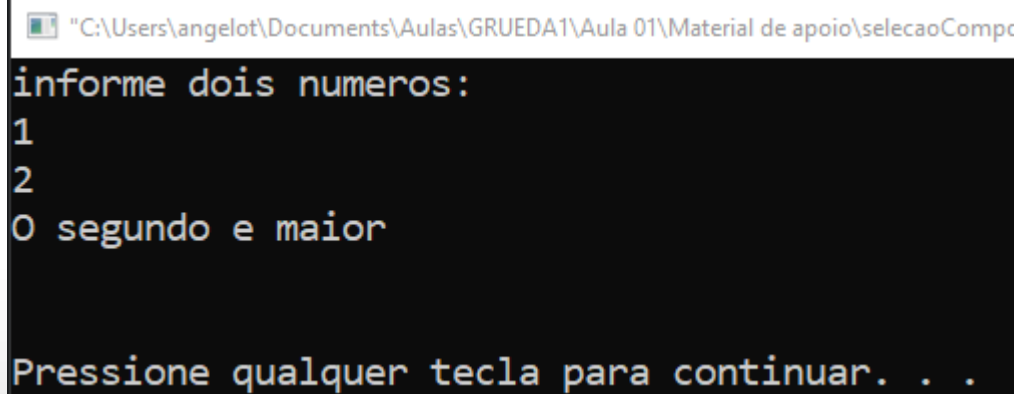
Estruturas de dados 1

Comandos de seleção

Seleção composta if - else - if - primeira forma de montagem do código

```
if('expressão'){  
    //blocos de comandos do if  
}else if('expressão'){  
    //bloco de comandos do 1º else-if  
}else if('expressão'){  
    //bloco de comandos do 2º else-if  
...  
}else if('expressão'){  
    //bloco de comandos do nº else-if  
}
```

```
#include <stdio.h>  
#include <stdlib.h>  
  
int main(){  
    int a, b;  
    printf("informe dois numeros: \n");  
    scanf(" %d %d", &a, &b);  
  
    if(a == b){  
        printf("Os numeros sao iguais \n\n\n");  
    }else{  
        if(a > b){  
            printf("O primeiro e maior \n\n\n");  
        }else{  
            printf("O segundo e maior \n\n\n");  
        }  
    }  
    system("pause");  
    return 0;  
}
```



```
"C:\Users\angelot\Documents\Aulas\GRUEDA1\Aula 01\Material de apoio\selecaoComp"  
informe dois numeros:  
1  
2  
O segundo e maior  
  
Pressione qualquer tecla para continuar. . .
```



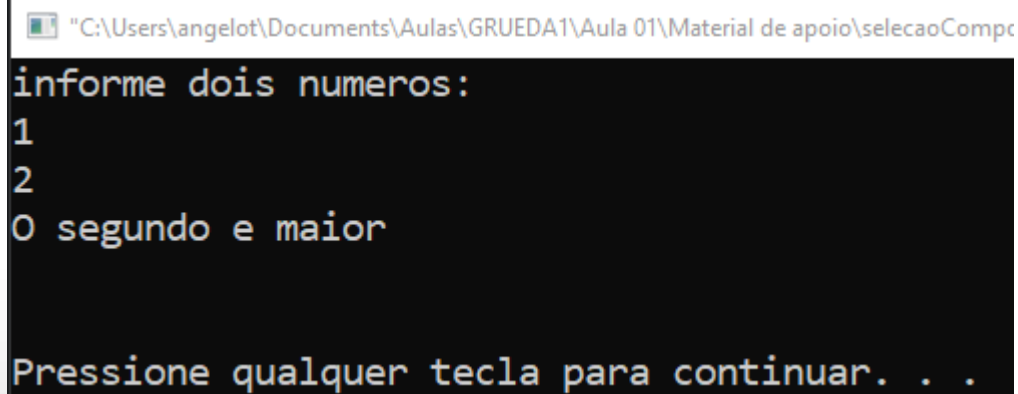
Estruturas de dados 1

Comandos de seleção

Seleção composta if - else - if - segunda forma de montagem do código

```
if('Expressão'){  
    ...  
}else{  
    if('expressão'){  
        ...  
    }else{  
        if('expressão'){  
            ...  
        }else{  
            //últimos comandos  
        }  
    }  
}
```

```
#include <stdio.h>  
#include <stdlib.h>  
  
int main(){  
    int a, b;  
    printf("informe dois numeros: \n");  
    scanf(" %d %d", &a, &b);  
  
    if(a == b){  
        printf("Os numeros sao iguais \n\n\n");  
    }else{  
        if(a > b){  
            printf("O primeiro e maior \n\n\n");  
        }else{  
            printf("O segundo e maior \n\n\n");  
        }  
    }  
    system("pause");  
    return 0;  
}
```



```
"C:\Users\angelot\Documents\Aulas\GRUEDA1\Aula 01\Material de apoio\selecaoComp  
informe dois numeros:  
1  
2  
O segundo e maior  
  
Pressione qualquer tecla para continuar. . .
```



Estruturas de dados 1

Comandos de seleção

Operador ternário - ?

Este operador, avalia a `expr1`, caso seja verdadeira, executa a `expr2`, senão, executa a `expr3`:

```
(expr1) ? (expr2) : (expr3);
```

```
//no exemplo ao lado
```

```
(a > 12) ? 100 : 200;
```

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

```
int main(){
    int a, b;
    printf("informe um numero: \n");
    scanf(" %d", &a);

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

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

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

```
"C:\Users\angelot\Documents\Aulas\GRUEDA1\Aula 01\Material de apoio\operadorTernar
informe um numero:
12
Valor de b: 200

Pressione qualquer tecla para continuar. . .
```



Estruturas de dados 1

Comandos de seleção

Seleção múltipla - switch

```
switch('expressao ou valor'){  
    case expr1:  
        //bloco de comandos expr1  
        break;  
    case expr2:  
        //bloco de comandos expr2  
        break;  
    ...  
    ...  
    default:  
        //bloco de comandos a executar  
        //quando nenhuma das opções é fornecida  
}  
}
```

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

```
int main(){  
    char escolha;  
    printf("Quer continuar? [sim - s / nao - n]: \n");  
    scanf(" %c", &escolha);  
  
    switch(escolha){  
        case 's': //aspas simples para caracteres!!  
            printf("Voce quer continuar");  
            break;  
        case 'n':  
            printf("Voce quer parar");  
            break;  
        default:  
            printf("Escolha invalida!");  
            break;  
    }  
    printf("\n\n\n");  
    system("pause");  
    return 0;  
}
```

"C:\Users\angelot\Documents\Aulas\GRUEDA1\Aula 01\Material de apoio\switch.exe"

Quer continuar? [sim - s / nao - n]:

s

Voce quer continuar

Pressione qualquer tecla para continuar. . .



Estruturas de dados 1

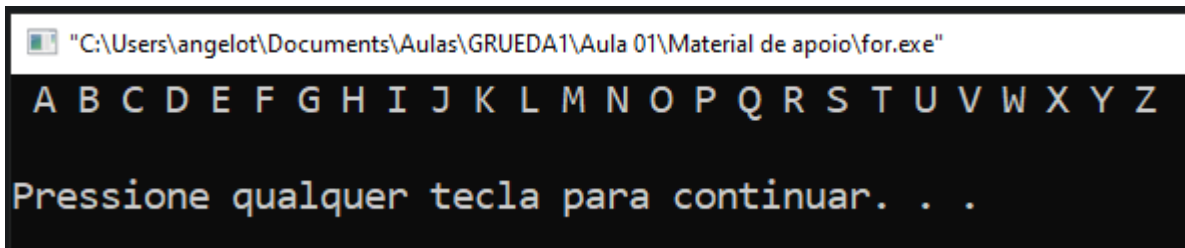
Comandos de iteração

O comando for

Executa repetidamente um bloco de comandos por um número definido de vezes

```
for('valor inicial'; 'condição de término'; 'incremento'){  
    //comandos a se repetirem um determinado numero de vezes  
}
```

```
#include <stdio.h>  
#include <stdlib.h>  
  
int main(){  
    int i;  
    char letra = 'A';  
    for(i = 0; i < 26; i++){  
        printf(" %c", letra);  
        letra++;  
    }  
  
    printf("\n\n");  
    system("pause");  
    return 0;  
}
```



```
"C:\Users\angelot\Documents\Aulas\GRUEDA1\Aula 01\Material de apoio\for.exe"  
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z  
Pressione qualquer tecla para continuar. . .
```



Estruturas de dados 1

Comandos de iteração

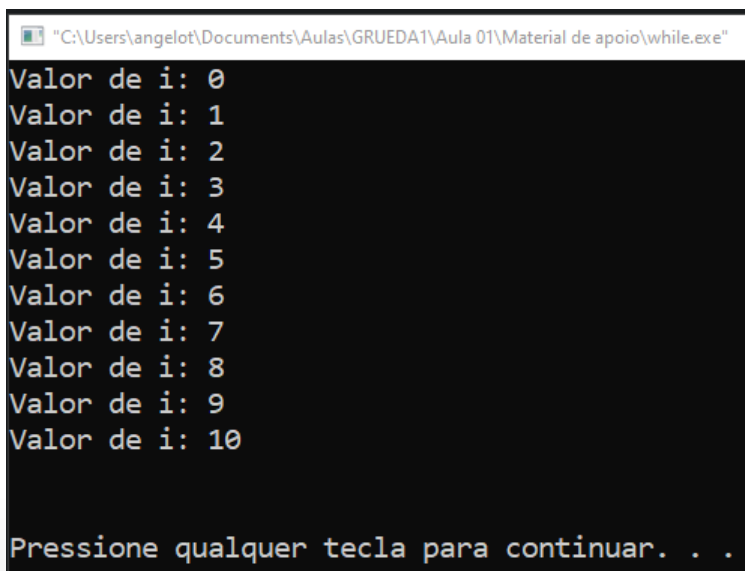
O comando while

Executa um bloco de comandos enquanto a condição avaliada for verdadeira

```
while('condição avaliada'){  
    //comandos a serem executados  
    //caso a condição seja verdadeira  
}
```

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

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



```
"C:\Users\angelot\Documents\Aulas\GRUEDA1\Aula 01\Material de apoio\while.exe"  
Valor de i: 0  
Valor de i: 1  
Valor de i: 2  
Valor de i: 3  
Valor de i: 4  
Valor de i: 5  
Valor de i: 6  
Valor de i: 7  
Valor de i: 8  
Valor de i: 9  
Valor de i: 10  
  
Pressione qualquer tecla para continuar. . .
```



Estruturas de dados 1

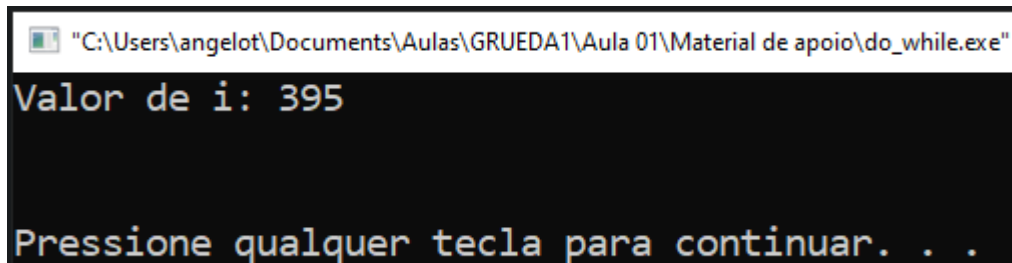
Comandos de iteração

O comando do/while

Executa um bloco de comandos, pelo menos uma vez, e avalia se a condição é verdadeira ao final do laço.

```
do{  
    //comandos a serem executados  
}while('condição avaliada');
```

```
#include <stdio.h>  
#include <stdlib.h>  
  
int main(){  
    int i = 395;  
  
    do{  
        printf("Valor de i: %d \n", i);  
        i++;  
    }while(i <= 10);  
  
    printf("\n\n");  
    system("pause");  
    return 0;  
}
```



"C:\Users\angelot\Documents\Aulas\GRUEDA1\Aula 01\Material de apoio\do_while.exe"

Valor de i: 395

Pressione qualquer tecla para continuar. . .



Estruturas de dados 1

Comandos de iteração

O comando continue

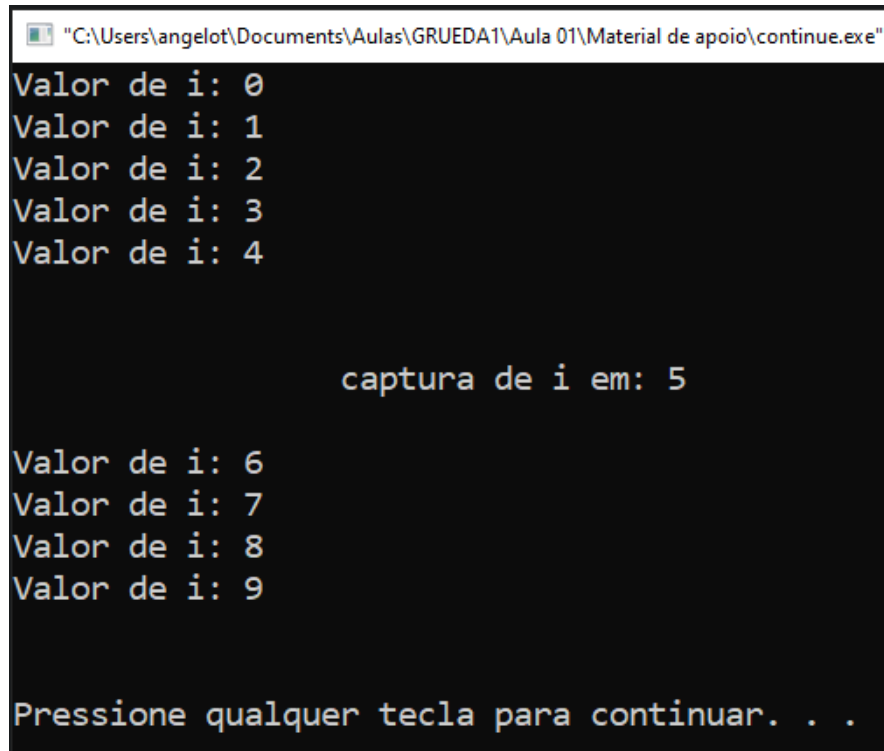
Ignora o restante das instruções abaixo do comando continue, dentro do laço, e obriga a execução a retornar para o início do laço, na próxima iteração:

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

int main(){
    int i;

    for(i = 0; i < 10; i++){
        if(i == 5){
            printf("\n\n\t\t captura de i em: %d \n\n", i);
            continue;
            printf("Observe que esta msg não será impressa!!");
        }
        printf("Valor de i: %d \n", i);
    }

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



```
"C:\Users\angelot\Documents\Aulas\GRUEDA1\Aula 01\Material de apoio\continue.exe"
Valor de i: 0
Valor de i: 1
Valor de i: 2
Valor de i: 3
Valor de i: 4

                                captura de i em: 5

Valor de i: 6
Valor de i: 7
Valor de i: 8
Valor de i: 9

Pressione qualquer tecla para continuar. . .
```



Estruturas de dados 1

Comandos de iteração

O comando break

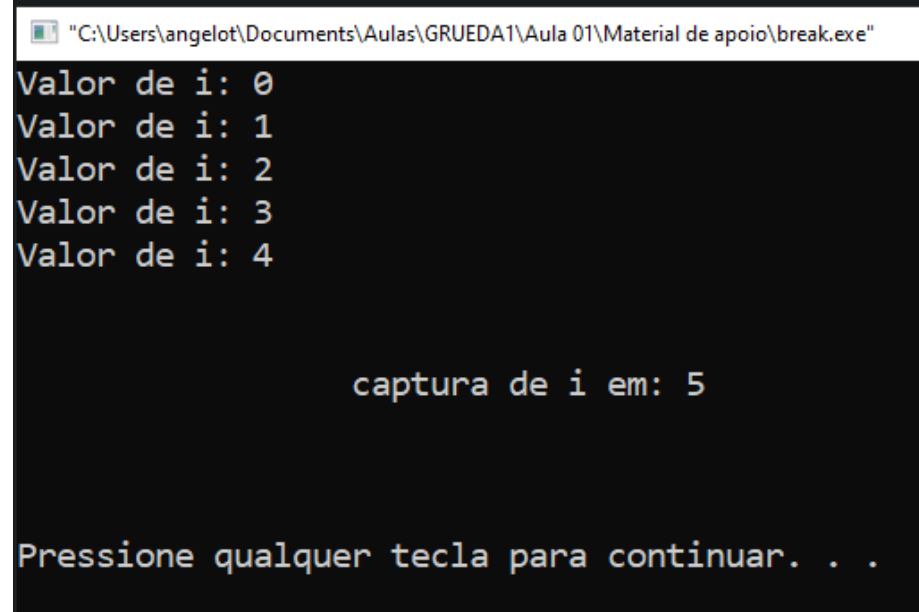
Encerra a execução dos comandos de laços for, while, do/while ou switch, forçando o desvio do fluxo do programa para a primeira linha de código após o bloco de código do laço, onde é utilizado:

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

int main() {
    int i;

    for(i = 0; i < 10; i++){
        if(i == 5){
            printf("\n\n\t\t captura de i em: %d \n\n", i);
            break;
            printf("Observe que esta msg não será impressa!!");
        }
        printf("Valor de i: %d \n", i);
    }

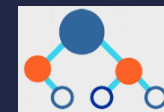
    printf("\n\n");
    system("pause");
    return 0;
}
```



```
"C:\Users\angelot\Documents\Aulas\GRUEDA1\Aula 01\Material de apoio\break.exe"
Valor de i: 0
Valor de i: 1
Valor de i: 2
Valor de i: 3
Valor de i: 4

captura de i em: 5

Pressione qualquer tecla para continuar. . .
```



Estruturas de dados 1

Comandos de iteração

O comando goto

Utilizado para saltar de um ponto para outro em um programa. Este comando está *depreciado*, ou seja, não é mais utilizado por questões de legibilidade do código. Existem outras formas e maneiras de codificação muito mais eficientes, que geram um código mais “limpo”. Só é mantido ainda, por questões de compatibilidade com programas legados.

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

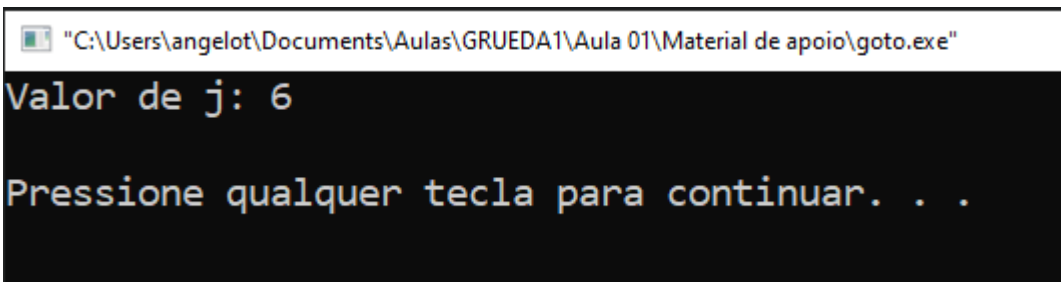
int main(){
    int j = 5;

    if(j == 5){
        j++;
        goto imprime;
    }

    j = 1350; //esta atribuição será ignorada

imprime:
    printf("Valor de j: %d", j);

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



```
"C:\Users\angelot\Documents\Aulas\GRUEDA1\Aula 01\Material de apoio\goto.exe"
Valor de j: 6
Pressione qualquer tecla para continuar. . .
```



Estruturas de dados 1

Atividade 1

- Codifique os exemplos apresentados do *slides* 16 ao 27, e verifique o funcionamento de todos e faça experimentos com valores diferentes onde for possível.
- Entregue somente os arquivos de código fonte (.c) compactados (zipados), na plataforma Moodle, como atividade 1.

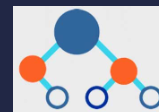


Estruturas de dados 1

Atividade 2

- Desenvolva 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 para a seleção o comando `switch`.
- Entregue no Moodle como atividade 2, somente o arquivo fonte (.c).

Entrada	Retorno
1	A
2	B
3	C
4	D



Estruturas de dados 1

Atividade 3

- Reescreva o programa anterior utilizando `if-else-if`.
- Entregue no Moodle como atividade 3, somente o arquivo fonte(.c).

Entrada	Retorno
1	A
2	B
3	C
4	D



Estruturas de dados 1



- No desenvolvimento de algoritmos, torna-se necessário estruturar os dados que serão manipulados pelo algoritmo, de forma que as operações mais comuns, possam ser executadas de forma mais eficiente. Para isso, os programadores contam com vários tipos de estruturas pré-definidas e estruturas que ainda podem ser definidas, inclusive pelo próprio programador, que assim, as utilizará com o objetivo de suprir a sua necessidade específica.
 - Vetores ou arranjo (array), Matrizes (Estruturas Homogêneas);
 - Registros ou `structs` (Estruturas Heterogêneas);
 - Listas (estática ou dinâmica);
 - Filas;
 - Pilhas;
 - Árvores;
 - Grafos;
 - etc.

Estruturas de dados 1

Estruturas de dados Homogêneas

- *Arrays*, vetores ou Matrizes:
 - São estruturas que podem armazenar vários valores ou dados **do mesmo tipo**. Podemos criar estes tipos de estruturas para todos os tipos primitivos da linguagem, e **também para os tipos que serão criados pelo programador**.
 - `int` – inteiros;
 - `char` – caracteres;
 - `float` – ponto flutuante;
 - `double` – ponto flutuante de dupla precisão;
 - etc.
 - Não é uma regra, mas convencionou-se informalmente chamarmos de vetor ou *array*, ou ainda arranjo, a matriz que possui apenas uma dimensão.



Estruturas de dados 1

Estruturas de dados Homogêneas

- Sintaxe para a 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] = {valor₁, valor₂, ..., valor_{qt}}

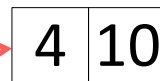
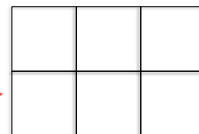
Exemplo

int notas[2];

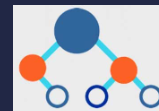
int matriz[2][3];

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

char palavra[4] = "LUA";



Matriz de 1 dimensão
sendo inicializada

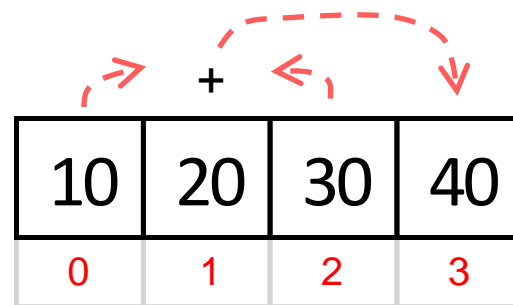


Estruturas de dados 1

Estruturas de dados Homogêneas

- Declaração e representação de uma matriz e acesso aos seus elementos:

```
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]);
```



Estruturas de dados 1

Estruturas de dados Homogêneas

- Exemplo de manipulação de uma matriz do tipo vetor:

```
#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]);
    }
```



Estruturas de dados 1

Estruturas de dados Homogêneas

```
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 posicao: %d e %d\n", i + 1 , j + 1);  
        }  
    }  
}  
system("pause");  
}
```

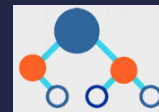
```
"C:\Users\angelot\Documents\Aulas\GRUEDA1\Aula 01\Material de apoio\vetor_valores_iguais.exe"  
  
Informe o valor do elemento 1 do vetor 1: 1  
Informe o valor do elemento 2 do vetor 1: 2  
Informe o valor do elemento 3 do vetor 1: 2  
Informe o valor do elemento 4 do vetor 1: 2  
Informe o valor do elemento 5 do vetor 1: 2  
Informe o valor do elemento 1 do vetor 2: 3  
Informe o valor do elemento 2 do vetor 2: 3  
Informe o valor do elemento 3 do vetor 2: 1  
Informe o valor do elemento 4 do vetor 2: 3  
Informe o valor do elemento 5 do vetor 2: 3  
  
Valores iguais na posicao: 1 e 3  
Pressione qualquer tecla para continuar. . .
```



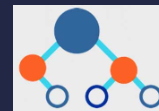
Estruturas de dados 1

Atividade prática 4

- Codifique o programa do exemplo anterior e teste seu funcionamento;
- Modifique-o para que imprima em tela os 2 vetores em linhas separadas identificando-os com seu nome na impressão. Em seguida exiba as posições em que possuem valores comuns ou se não possuem (dica: use uma variável para indicar se houve coincidência);
- Entregue somente o arquivo fonte (.c) na plataforma Moodle, na entrada “Atividade 4”.



Estruturas de dados 1



Exemplo de manipulação de uma matriz 5 x 5

```
#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]);
        }
    }
```



Estruturas de dados 1

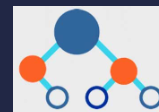


Exemplo de manipulação de uma matriz 5 x 5

- 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;  
}  
↓
```

Estruturas de dados 1



Exemplo de manipulação de uma matriz 5 x 5

- 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;  
}  
↓
```


Estruturas de dados 1



Exemplo de manipulação de uma matriz 5 x 5

- Exibindo os valores totalizados



```
//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]);
}

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

Estruturas de dados 1

Atividade prática 5

- Codifique o programa matrizes 5 x 5, em todas as suas etapas exemplificada anteriormente;
- Complete o programa adicionando um bloco de código para a rotina de impressão afim de visualizar a matriz no console (tela do terminal). Este bloco deve imprimir o conteúdo da matriz em tela, no formato de uma matriz, ou seja, com linhas e colunas;
- Entregue o arquivo fonte (.c) na plataforma Moodle, na entrada atividade 5.

