

Teoria da Computação & Compiladores

Wesley Dias Maciel: wesley.dias@prof.una.br



C

Linguagem de Programação C

C



Compiladores

<http://www.codeblocks.org/>



Code::Blocks

<http://www.bloodshed.net/devcpp.html>



<https://github.com/>



<https://www.gitpod.io/>



Gitpod

Compiladores



Code::Blocks

<http://www.codeblocks.org/>

<https://www.codeblocks.org/downloads/binaries/#imagesoswindows48pnglogo-microsoft-windows>

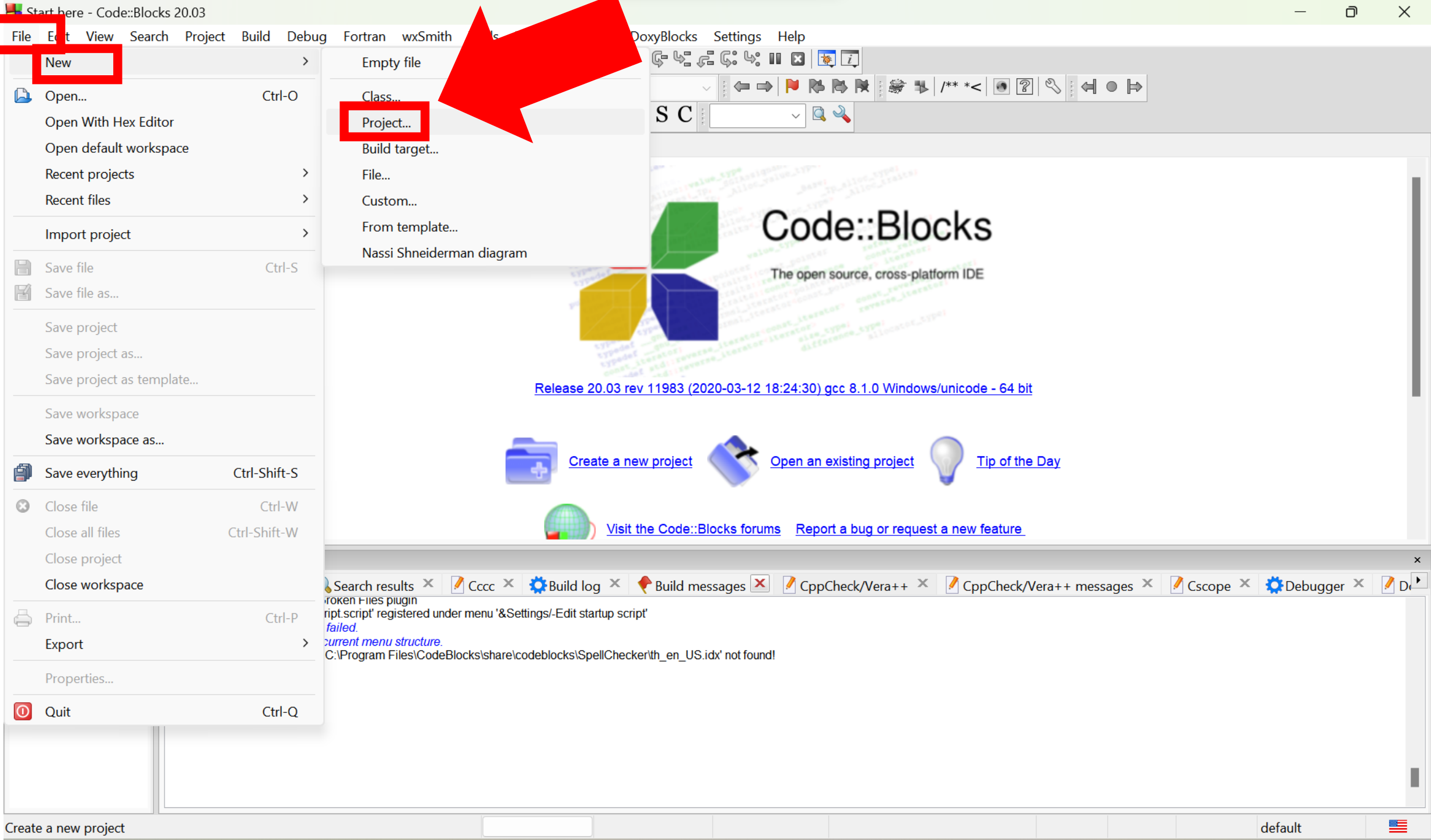


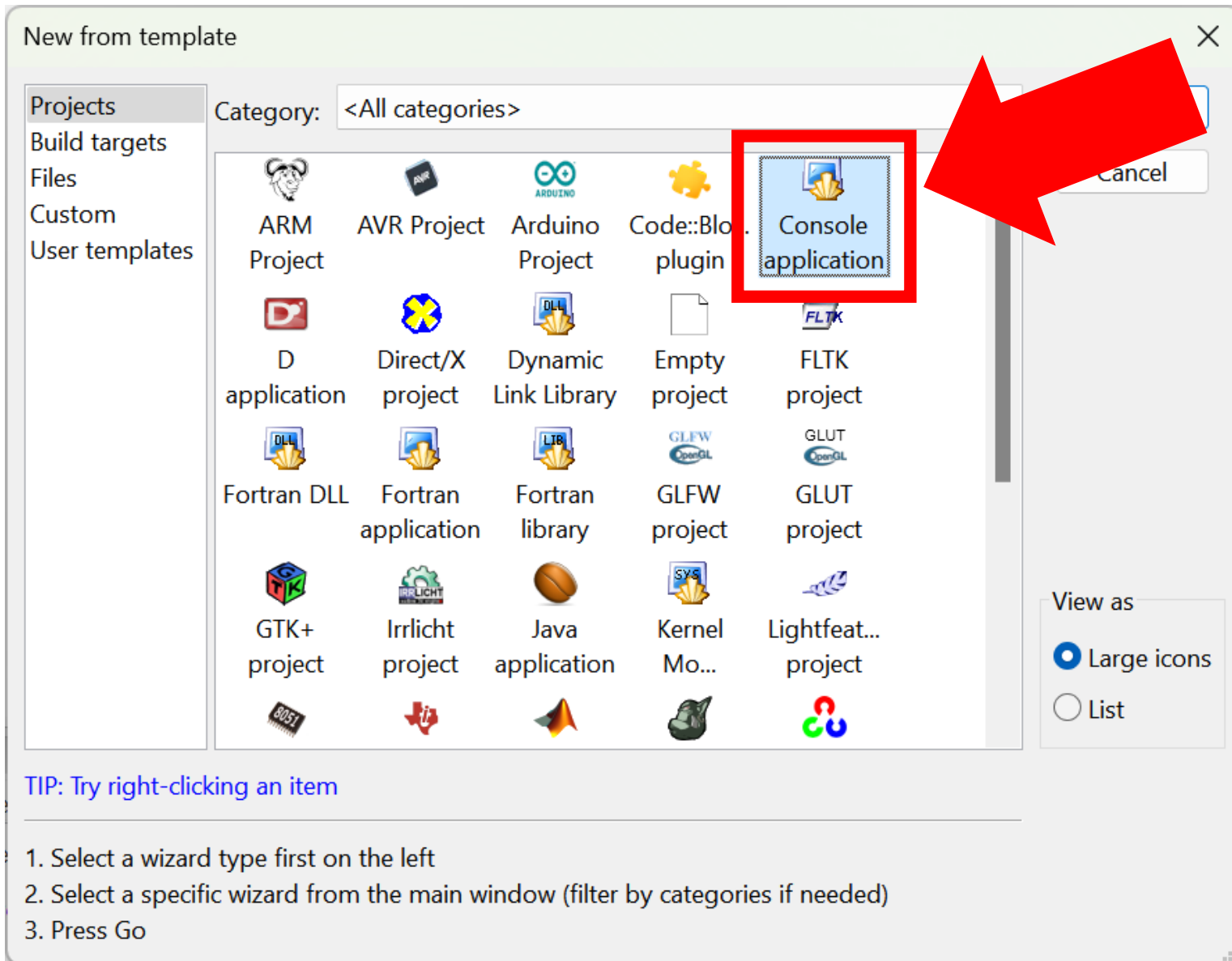
Microsoft Windows

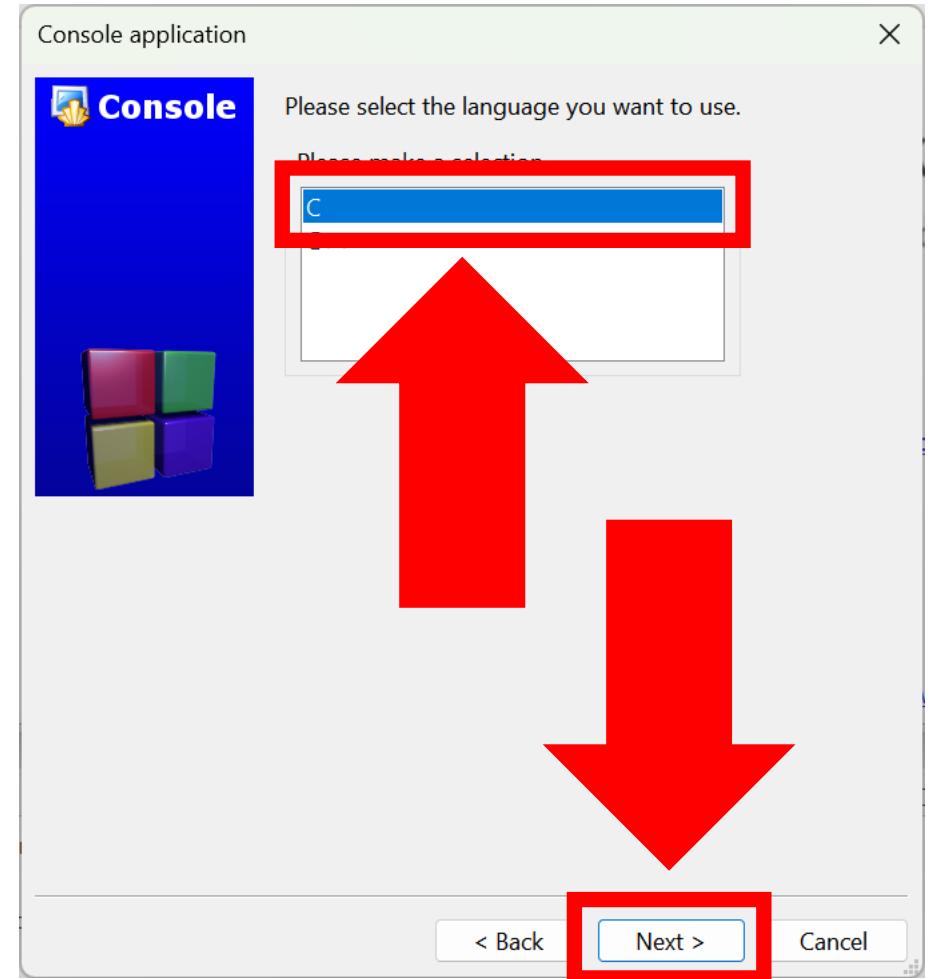
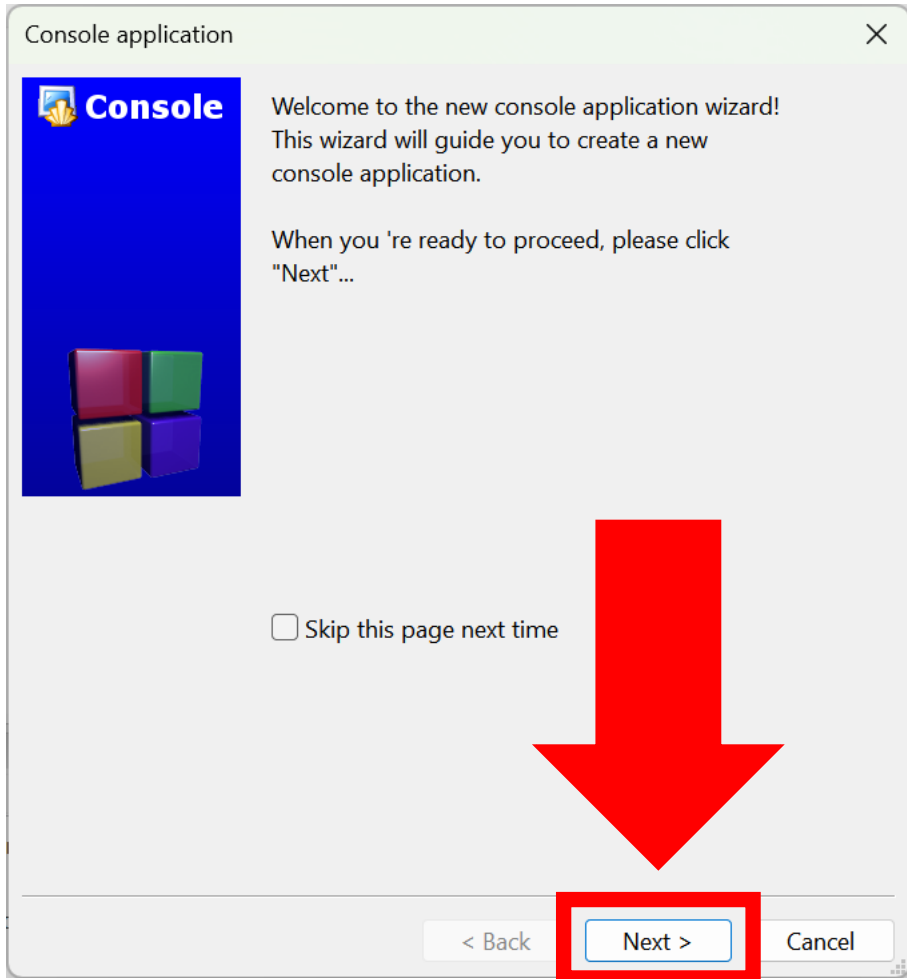
	File	Download from
	codeblocks-20.03-setup.exe	FossHUB or Sourceforge.net
	codeblocks-20.03-setup-nonadmin.exe	FossHUB or Sourceforge.net
	codeblocks-20.03-nosetup.zip	FossHUB or Sourceforge.net
1	codeblocks-20.03mingw-setup.exe	FossHUB or Sourceforge.net
2	codeblocks-20.03mingw-nosetup.zip	FossHUB or Sourceforge.net
	codeblocks-20.03-32bit-setup.exe	FossHUB or Sourceforge.net
	codeblocks-20.03-32bit-setup-nonadmin.exe	FossHUB or Sourceforge.net
	codeblocks-20.03-32bit-nosetup.zip	FossHUB or Sourceforge.net
	codeblocks-20.03mingw-32bit-setup.exe	FossHUB or Sourceforge.net
	codeblocks-20.03mingw-32bit-nosetup.zip	FossHUB or Sourceforge.net

NOTE: The codeblocks-20.03-setup.exe file includes Code::Blocks with all plugins. The codeblocks-20.03-setup-nonadmin.exe file is provided for convenience to users that do not have administrator rights on their machine(s).


NOTE: The codeblocks-20.03mingw-setup.exe file includes additionally the GCC/G++/GFortran compiler and GDB debugger from [MinGW-W64 project](#) (version 8.1.0, 32/64 bit, SEH).







Console application

 Please select the folder where you want the new project to be created as well as its title.


Project title:
exemplo

Folder to create project in:
C:\temp\linguagem-C

Project filename:
exemplo.cbp


Resulting filename:
C:\temp\linguagem-C\exemplo\exemplo.cbp

preencher



< Back Next > Cancel

Console application

 Please select the compiler to use and which configurations you want enabled in your project.

Compiler:
GNU GCC Compiler


☒ Create "Debug" configuration: Debug

"Debug" options
Output dir.: bin\Debug
Objects output dir.: obj\Debug

☒ Create "Release" configuration: Release

"Release" options
Output dir.: bin\Release
Objects output dir.: obj\Release

< Back Finish Cancel



main.c [exemplo] - Code::Blocks 20.03

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

<global>

Management

Projects Files

Workspace

exemplo

Sources

main.c

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main()
5 {
6     printf("Hello world!\n");
7     return 0;
8 }
9
```

Logs & others

Code::Blocks Search results Cccc Build log Build messages CppCheck/Vera++ CppCheck/Vera++ messages Cscope Debugger D

wxSmithInit
wxSmithAui
wxSmithContribItems
WindowsXPLookAndFeel
Initial scaling factor is 1.500 (actual: 1.500)
Running startup script
Script plugin registered: Find Broken Files plugin
Script/function 'edit_startup_script' registered under menu '&Settings/-Edit startup script'
Preserving older key bindings failed.
Will create key bindings from current menu structure.
SpellChecker: Thesaurus files 'C:\Program Files\CodeBlocks\share\codeblocks\SpellChecker\th_en_US.idx' not found!
ProjectManager::SetProject took: 0.062 seconds.
NativeParser::DoFullParsing took: 2.118 seconds.
NativeParser::CreateParser: Finish creating a new parser for project 'exemplo'
NativeParser::OnParserEnd: Project 'exemplo' parsing stage done!

C:\temp\linguagem-C\exemplo\main.c C/C++ Windows (CR+LF) WINDOWS-1252 Line 9, Col 1, Pos 107 Insert Read/Write default

Compiladores e Interpretadores Online

https://www.tutorialspoint.com/compile_c_online.php



tutorialspoint | **Online C Compiler**

Project ▾ Edit ▾ Setting ▾ [-> Wesley ▾

Execute | Beautify | Share | Source Code | Help

```
1  /* Online C Compiler and Editor */
2  #include <stdio.h>
3
4  int main()
5  {
6      int num;
7      printf("Informe um número: ");
8      scanf("%d", &num);
9      printf("\nNúmero informado: %d", num);
10     return 0;
11 }
```

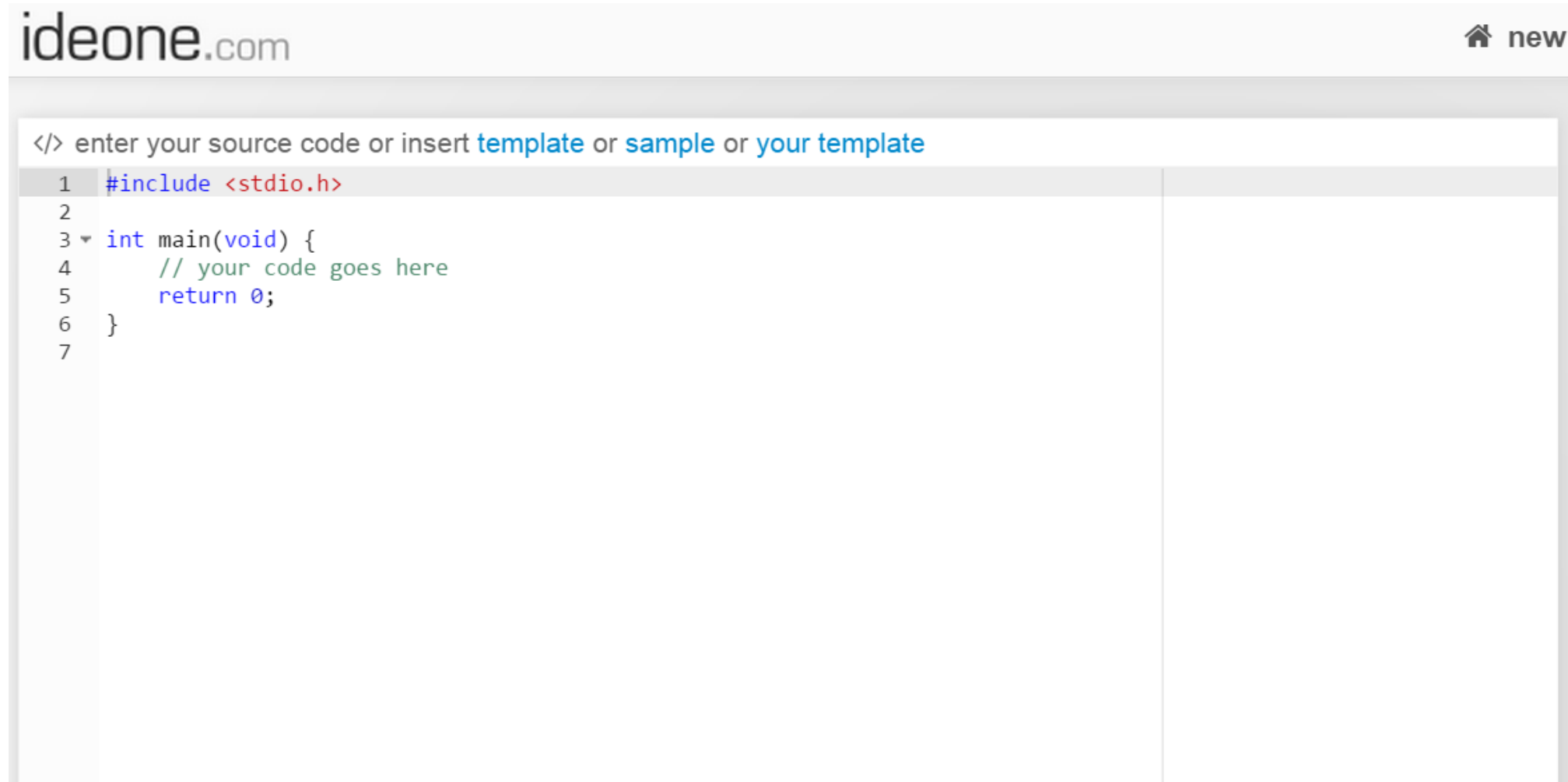
Terminal

Informe um número: 17
Número informado: 17

Advertisement

Compiladores e Interpretadores Online

<https://ideone.com/>



The screenshot shows the ideone.com website interface. At the top left is the logo "ideone.com". At the top right is a home icon and the word "new". Below the header is a text input area with the placeholder text "</> enter your source code or insert [template](#) or [sample](#) or [your template](#)". Below this is a code editor with a light gray background and a vertical line on the right. The code editor contains the following C code:

```
1 #include <stdio.h>
2
3 int main(void) {
4     // your code goes here
5     return 0;
6 }
7
```

Comando de Saída - Monitor

```
#include <stdio.h> //printf().
```

```
int main() {  
    printf("\nOla Mundo!!! \n\n\n");  
    return 0;  
}
```

Declaração de Variáveis - Atribuição

```
#include <stdio.h> //printf().
```

```
int main() {
```

```
    int x;
```

```
    x = 65;
```

```
    printf("Valor de x: %d \n\n\n", x); //O "%d" pode ser substituído por "%i".
```

```
    return 0;
```

```
}
```

OBS: se o "%d" for substituído por "%c", será apresentado o caracter "A" (correspondente ao código **65** na tabela **ASC II**) .

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	\$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[END OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

<https://pt.m.wikipedia.org/wiki/Ficheiro:ASCII-Table-wide.svg>

Tipos Primitivos

- Existem 5 tipos de dados primitivos (pré-definidos) em C:

Palavra Chave	Tipo
char	character
int	inteiro
float	real de precisão simples
double	real de precisao dupla
void	vazio (sem valor)

Tipos Primitivos

- Com exceção de float e void, os outros tipos de dados primitivos podem ter modificadores.
- Ao tipo primitivo double pode-se aplicar apenas o modificador long.
- Os modificadores alteram o tamanho do tipo de dado ou sua forma de representação.

Palavra Chave	Tipo
signed	caracter
unsigned	Inteiro
long	longo
short	curto

Tipos Primitivos

- Os quatro modificadores podem ser aplicados a inteiros.
- A intenção é que **short** e **long** proporcionem tamanhos diferentes de inteiros onde isto for conveniente.
- **int** normalmente terá o tamanho natural para uma determinada máquina (tamanho da palavra da máquina). Assim:
 - Numa máquina de 16 bits, **int** provavelmente terá 16 bits.
 - Numa máquina de 32, deverá ter 32 bits.
- Na verdade, cada compilador é livre para escolher tamanhos adequados para o seu próprio hardware.

Tipos Primitivos

Lista dos tipos de dados primitivos e seu valores mínimos e máximos em um compilador típico para um hardware de **16 bits**:

Tipo	Num de bits	Intervalo	
		Inicio	Fim
char	8	-128	127
unsigned char	8	0	255
signed char	8	-128	127
int	16	-32.768	32.767
unsigned int	16	0	65.535
signed int	16	-32.768	32.767
short int	16	-32.768	32.767
unsigned short int	16	0	65.535
signed short int	16	-32.768	32.767
long int	32	-2.147.483.648	2.147.483.647
signed long int	32	-2.147.483.648	2.147.483.647
unsigned long int	32	0	4.294.967.295
float	32	3,4E-38	3.4E+38
double	64	1,7E-308	1,7E+308
long double	80	3,4E-4932	3,4E+4932

sizeof()

```
int main() {  
    int x;  
  
    printf ("\nTamanhos: ");  
    printf ("\nx: %d bytes", sizeof(x));  
    printf ("\n\nshort int: %d bytes", sizeof(short int));  
    printf ("\n\nint: %d bytes", sizeof(int));  
    printf ("\n\nlong int: %d bytes", sizeof(long int));  
    printf ("\n\nfloat: %d bytes", sizeof(float));  
    printf ("\n\ndouble: %d bytes\n\n", sizeof(double));  
    return 0;  
}
```

Tamanhos:

x: 4 bytes

short int: 2 bytes

int: 4 bytes

long int: 4 bytes

float: 4 bytes

double: 8 bytes

Comando de Entrada - Teclado

```
#include <stdio.h> //printf(), scanf().
```

```
int main() {  
    int x;  
  
    printf ("\nInforme um numero: ");  
    scanf ("%d", &x);  
    printf("Valor de x: %d \n\n\n", x);  
    return 0;  
}
```

Lendo um Character após uma Execução Qualquer de scanf()

```
#include <stdio.h> //printf(), scanf().
```

```
int main() {
```

```
    int num;
```

```
    char lixo, letra;
```

```
    printf ("\nInforme um numero: ");
```

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

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

```
    //scanf ("%c", &lixo); //Esta linha é obrigatória (remover o comentário, para que o programa execute corretamente).
```

```
    printf ("\nInforme uma letra: ");
```

```
    scanf ("%c", &letra);
```

```
    printf("Letra informada: %c \n\n\n", letra);
```

```
    return 0;
```

```
}
```

Operadores Binários

- Adição: +
 - Exemplo: `resp = num1 + num2;`
- Subtração: -
 - Exemplo: `resp = num1 - num2;`
- Multiplicação:
 - Exemplo: `resp = num1 * num2;`
- Divisão inteira: /
 - Exemplo: `resp = num1 / num2;`
- Resto (módulo): %
 - Exemplo: `resp = num1 % num2;`

Operadores Binários

```
int main() {  
    int num1 = 3, num2 = 5;  
  
    printf("\nnum1 = %d", num1);  
    printf("\nnum2 = %d", num2);  
    printf("\nnum1 + num2 = %d", num1 + num2);  
    printf("\nnum1 - num2 = %d", num1 - num2);  
    printf("\nnum1 * num2 = %d", num1 * num2);  
    printf("\nnum1 / num2 = %d", num1 / num2);  
    printf("\nnum1 %% num2 = %d", num1 % num2);  
    return 0;  
}
```

```
num1 = 3  
num2 = 5  
num1 + num2 = 8  
num1 - num2 = -2  
num1 * num2 = 15  
num1 / num2 = 0  
num1 % num2 = 3
```


Operadores Binários

Divisão não inteira: (type cast) / (type cast)

- Exemplos:
 - `resp = (double) num1 / num2;`
 - `resp = num1 / (double) num2;`
 - `resp = (double) num1 / (double) num2;`
- **OBS:** só é necessária quando os dois operadores forem inteiros.

Operadores Binários

Divisão não inteira: (type cast) / (type cast)

```
int main() {  
    int num1 = 3, num2 = 5;  
  
    printf("\nnum1 = %d", num1);  
    printf("\nnum2 = %d", num2);  
    printf("\n(double) num1 / num2 = %f", (double) num1 / num2);  
    printf("\num1 / (double) num2 = %f", num1 / (double) num2);  
    printf("\n(double) num1 / (double) num2 = %f\n", (double) num1 / (double) num2);  
    return 0;  
}
```

```
num1 = 3  
num2 = 5  
(double) num1 / num2 = 0.600000  
num1 / (double) num2 = 0.600000  
(double) num1 / (double) num2 = 0.600000
```

Operadores de Atribuição

- Atribuição: =
 - Exemplo: `int resp = 10, num = 5;`
- Adição: +=
 - Exemplo: `resp += num;`
- Subtração: -=
 - Exemplo: `resp -= num;`
- Multiplicação:
 - Exemplo: `resp *= num;`
- Divisão inteira: /=
 - Exemplo: `resp /= num;`
- Resto (módulo): %
 - Exemplo: `resp %= num;`

Operadores de Atribuição

```
int main() {  
    int resp = 10, num = 5;  
  
    resp += num;  
    printf("\nresp += num: %d", resp);  
    resp -= num;  
    printf("\nresp -= num: %d", resp);  
    resp *= num;  
    printf("\nresp *= num: %d", resp);  
    resp /= num;  
    printf("\nresp /= num: %d", resp);  
    resp %= num;  
    printf("\nresp %= num: %d", resp);  
    return 0;  
}
```

```
resp += num: 15  
resp -= num: 10  
resp *= num: 50  
resp /= num: 10  
resp %= num: 0
```

Operadores Unários Pré-fixados

- Interpretação: “executa depois usa”:
- Adição: ++
 - Exemplo: `resp = ++num;`
- Subtração: --
 - Exemplo: `resp = --num;`

Operadores Unários Pré-fixados

```
int main() {  
    int num = 5;  
    printf("\n%d", num); //5  
    printf("\n%d", ++num); //6  
    printf("\n%d", num); //6  
    printf("\n%d", --num); //5  
    printf("\n%d", num); //5  
  
    return 0;  
}
```

Operadores Unários Pós-fixados

- Interpretação: “usa depois executa”:
- Adição: ++
 - Exemplo: `resp = num++;`
- Subtração: --
 - Exemplo: `resp = num--;`

Operadores Unários Pós-fixados

```
int main() {  
    int num = 5;  
    printf("\n%d", num); //5  
    printf("\n%d", num++); //5  
    printf("\n%d", num); //6  
    printf("\n%d", num--); //6  
    printf("\n%d", num); //5  
  
    return 0;  
}
```


Operadores Relacionais

Retornam: 0 (**falso**) ou 1 (**verdadeiro**)

- Maior: >
 - Exemplo: `resp = 5 > 2;`
- Maior ou igual: >=
 - Exemplo: `resp = 5 >= 2;`
- Menor: <
 - Exemplo: `resp = 5 < 2;`
- Menor ou igual: <=
 - Exemplo: `resp = 5 <= 2;`
- Diferente: !=
 - Exemplo: `resp = 5 != 2;`

Operadores Relacionais

```
int main() {  
  
    printf("\n%d", 5 > 2); //1  
    printf("\n%d", 5 >= 2); //1  
    printf("\n%d", 5 < 2); //0  
    printf("\n%d", 5 <= 2); //0  
    printf("\n%d", 5 != 2); //1  
  
    return 0;  
}
```



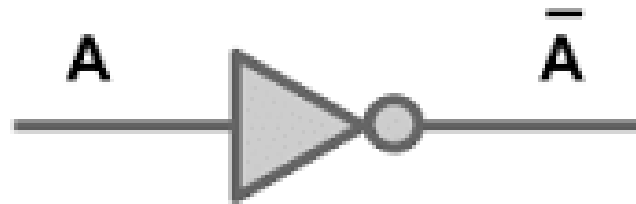
1
1
0
0
1

Operadores Lógicos

Retornam: 0 (falso) ou 1 (verdadeiro)

- NÃO (NOT): !
 - Exemplo: $\text{resp} = !(7 > 3)$

PORTA NÃO (NOT)



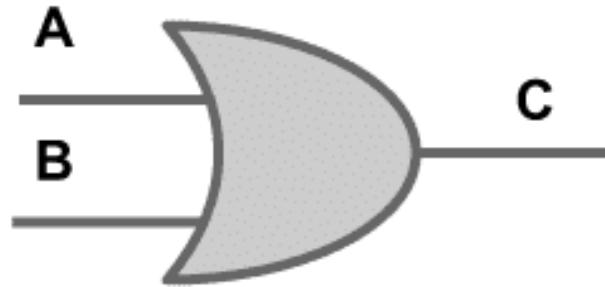
A	\bar{A}
0	1
1	0

Operadores Lógicos

Retornam: 0 (**falso**) ou 1 (**verdadeiro**)

- OU (OR): `||`
 - Exemplo: `resp = (2 > 5) || (7 > 3)`

PORTA OU (OR)



$C=A+B$

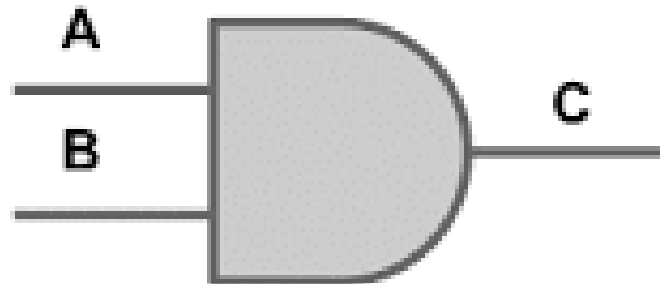
A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

Operadores Lógicos

Retornam: 0 (**falso**) ou 1 (**verdadeiro**)

- E (AND): &&
 - Exemplo: $\text{resp} = (5 > 2) \ \&\& \ (7 > 3)$

PORTA E (AND)



$$C = A \cdot B$$

A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

Operadores Lógicos

```
int main() {  
    printf("\n%d", !(7 > 3)); //0  
    printf("\n%d", (2 > 5) || (7 > 3)); //1  
    printf("\n%d", (5 > 2) && (7 > 3)); //1  
  
    return 0;  
}
```



0
1
1

Biblioteca: Funções Matemáticas

```
#include <stdio.h> //printf (), scanf ().
#include <math.h> //pow ().

int main() {
    int a, b;

    printf ("\nInforme a base: ");
    scanf ("%d", &a);
    printf ("\nInforme o expoente: ");
    scanf ("%d", &b);
    printf ("%d ^ %d = %f \n\n\n", a, b, pow (a, b));
    return 0;
}
```

```
#include <stdio.h> //printf (), scanf ().
#include <math.h> //pow ().

int main() {
    int a, b;

    printf ("\nInforme a base: ");
    scanf ("%d", &a);
    printf ("\nInforme o expoente: ");
    scanf ("%d", &b);
    printf ("%d ^ %d = %.2f \n\n\n", a, b, pow (a, b));
    return 0;
}
```

Formatação: 2 casas decimais.

Estrutura Condicional: If - Else

```
#include <stdio.h> //printf (), scanf ().
```

```
int main() {  
    int num1, num2;  
  
    printf ("\nInforme o primeiro numero: ");  
    scanf ("%d", &num1);  
    printf ("\nInforme o segundo numero: ");  
    scanf ("%d", &num2);  
    if (num1 > num2)  
        printf ("\nO maior numero informado e: %d", num1);  
    else  
        printf ("\nO maior numero informado e: %d", num2);  
    printf ("\n\n\n");  
    return 0;  
}
```


Estrutura de Decisão: switch-case

```
#include <stdio.h> //printf (), scanf ().
```

```
int main() {  
    int num;  
  
    printf ("\nInforme: ");  
    printf ("\n[1] Arroz: ");  
    printf ("\n[2] Fruta: ");  
    printf ("\n[3] Leite: \n");  
    scanf ("%d", &num);
```

```
    switch(num){  
        case 1:  
            printf("\nArroz: R$ 5,67 o kg");  
            break;  
        case 2:  
            printf("\nFruta: R$ 4,89 o kg");  
            break;  
        case 3:  
            printf("\nLeite: R$ 4,17 o litro");  
            break;  
        default:  
            printf("\nProduto nao identificado!");  
    }  
    printf ("\n\n\n");  
    return 0;  
}
```

Estrutura de Decisão: switch-case

```
#include <stdio.h> //printf (), scanf ().
```

```
int main() {  
    char letra;  
  
    printf ("\nInforme: ");  
    printf ("\n[a] Arroz: ");  
    printf ("\n[f] Fruta: ");  
    printf ("\n[l] Leite: \n");  
    scanf ("%c", &letra);
```

```
    switch((int) letra){  
        case 97:  
            printf("\nArroz: R$ 5,67 o kg");  
            break;  
        case 102:  
            printf("\nFruta: R$ 4,89 o kg");  
            break;  
        case 108:  
            printf("\nLeite: R$ 4,17 o litro");  
            break;  
        default:  
            printf("\nProduto nao identificado!");  
    }  
    printf ("\n\n\n");  
    return 0;  
}
```

Estrutura de repetição: while

```
#include <stdio.h> //printf().
```

```
int main() {  
    int i = 65;  
  
    while(i < 91) {  
        printf ("%c ", i);  
        i++;  
    }  
    printf ("\n\n\n");  
    return 0;  
}
```

Estrutura de repetição: do-while

```
#include <stdio.h> //printf().
```

```
int main() {  
    int i = 65;  
  
    do {  
        printf ("%c ", i);  
        i++;  
    } while(i < 91);  
    printf ("\n\n\n");  
    return 0;  
}
```

Estrutura de repetição: for

```
#include <stdio.h> //printf().
```

```
int main() {  
    int i;  
  
    for(i = 65; i < 91; i++)  
        printf ("%c ", i);  
  
    printf ("\n\n\n");  
    return 0;  
}
```

Procedimento

```
#include <stdio.h> //printf (), scanf ().

void maior () {
    int num1, num2;
    printf ("\nInforme o primeiro numero: ");
    scanf ("%d", &num1);
    printf ("\nInforme o segundo numero: ");
    scanf ("%d", &num2);
    if (num1 > num2)
        printf ("\nO maior numero informado e: %d", num1);
    else
        printf ("\nO maior numero informado e: %d", num2);
    printf ("\n\n\n");
}

int main() {
    maior ();
    return 0;
}
```

```
#include <stdio.h> //printf (), scanf ().

void maior (); //Protótipo ou assinatura.

int main() {
    maior ();
    return 0;
}

void maior () {
    int num1, num2;
    printf ("\nInforme o primeiro numero: ");
    scanf ("%d", &num1);
    printf ("\nInforme o segundo numero: ");
    scanf ("%d", &num2);
    if (num1 > num2)
        printf ("\nO maior numero informado e: %d", num1);
    else
        printf ("\nO maior numero informado e: %d", num2);
    printf ("\n\n\n");
}
```

Passagem de Parâmetro – Estrutura de Repetição: While

```
#include <stdio.h> //printf (), scanf ().
```

```
void pares (int num); //Protótipo ou assinatura.
```

```
int main() {  
    int num;  
    printf ("\nInforme um numero: ");  
    scanf ("%d", &num);  
    pares (num);  
    printf ("\n\n\n");  
    return 0;  
}
```

```
void pares (int n) {  
    int i = 0;  
    printf ("\nPares menores que %d:", n);  
    while (i < n) {  
        if (i % 2 == 0)  
            printf ("\n%d ", i);  
        i++;  
    }  
}
```

Passagem de Parâmetro – Estrutura de Repetição: Do - While

```
#include <stdio.h> //printf (), scanf ().
```

```
void pares (int num); //Protótipo ou assinatura.
```

```
int main() {  
    int num;  
    printf ("\nInforme um numero: ");  
    scanf ("%d", &num);  
    pares (num);  
    printf ("\n\n\n");  
    return 0;  
}
```

```
void pares (int n) {  
    int i = 0;  
    printf ("\nPares menores que %d:", n);  
    do {  
        if (i % 2 == 0)  
            printf ("\n%d ", i);  
        i++;  
    } while (i < n);  
}
```


Passagem de Parâmetro – Estrutura de Repetição: For

```
#include <stdio.h> //printf (), scanf ().
```

```
void pares (int num); //Protótipo ou assinatura.
```

```
int main() {
```

```
    int num;
```

```
    printf ("\nInforme um numero: ");
```

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

```
    pares (num);
```

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

```
    return 0;
```

```
}
```

```
void pares (int n) {
```

```
    int i;
```

```
    printf ("\nPares menores que %d:", n);
```

```
    for (i = 0; i < n; i++)
```

```
        if (i % 2 == 0)
```

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

```
}
```

Função

```
#include <stdio.h> //printf (), scanf ().
```

```
int maior (int n1, int n2); //Protótipo ou assinatura.
```

```
int main() {  
    int num1, num2, resp;  
    printf ("\nInforme o primeiro numero: ");  
    scanf ("%d", &num1);  
    printf ("\nInforme o segundo numero: ");  
    scanf ("%d", &num2);  
    resp = maior (num1, num2);  
    printf ("\nMaior numero informado: %d", resp);  
    printf ("\n\n\n");  
    return 0;  
}
```

```
int maior (int n1, int n2) {  
    int r = n1;  
  
    if (n2 > n1)  
        r = n2;  
  
    return r;  
}
```

Ponteiro

```
#include <stdio.h> //printf ().

int main() {
    int num = 5, *pt;

    pt = &num; //pt recebe o endereço de num.

    printf ("\n num: %d", num); //imprimir o conteúdo de num.
    printf ("\n &num: %p", &num); //imprimir o endereço de num.
    printf ("\n\n");

    printf ("\n *pt: %d", *pt); //imprimir o conteúdo da posição de memória para onde pt aponta.
    printf ("\n pt: %p", pt); //imprimir o conteúdo de pt (que é o endereço de num).
    printf ("\n &pt: %p", &pt); //imprimir o endereço de pt.
    printf ("\n\n");
    return 0;
}
```

Ponteiro para Ponterio

```
#include <stdio.h> //printf ().
```

```
int main() {
```

```
    int num = 65, *pt, **ptpt;
```

```
    pt = &num; //pt recebe o endereço de num.
```

```
    ptpt = &pt; //ptpt recebe o endereço de pt.
```

```
    printf ("\n num: %d", num); //imprimir o conteúdo de num - 65.
```

```
    printf ("\n &num: %p", &num); //imprimir o endereço de num.
```

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

```
    printf ("\n *pt: %d", *pt); //imprimir o conteúdo da posição de memória para onde pt aponta (que é o conteúdo de num - 65).
```

```
    printf ("\n pt: %p", pt); //imprimir o conteúdo de pt (que é o endereço de num).
```

```
    printf ("\n &pt: %p", &pt); //imprimir o endereço de pt.
```

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

Ponteiro para Ponteiro

```
printf ("\n **ptpt: %d", **ptpt); //imprimir o conteúdo da posição de memória para onde o ponteiro armazenado em ptpt aponta (que é o conteúdo de num - 65).  
printf ("\n *ptpt: %p", *ptpt); //imprimir o conteúdo da posição de memória para onde ptpt aponta (que é o conteúdo de pt – endereço de num).  
printf ("\n ptpt: %p", ptpt); //imprimir o conteúdo de ptpt (que é o endereço de pt).  
printf ("\n &ptpt: %p", &ptpt); //imprimir o endereço de ptpt.  
printf ("\n\n");  
  
return 0;  
}
```

Passagem de Ponteiro como Parâmetro

```
#include <stdio.h> //printf (), scanf ().
```

```
void maior (int n1, int n2, int *r); //Protótipo ou assinatura.
```

```
int main() {
```

```
    int num1, num2, resp;
```

```
    printf ("\nInforme o primeiro numero: ");
```

```
    scanf ("%d", &num1);
```

```
    printf ("\nInforme o segundo numero: ");
```

```
    scanf ("%d", &num2);
```

```
    maior (num1, num2, &resp);
```

```
    printf ("\nMaior numero informado: %d", resp);
```

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

```
    return 0;
```

```
}
```

```
void maior (int n1, int n2, int *r) {
```

```
    *r = n1;
```

```
    if (n2 > n1)
```

```
        *r = n2;
```

```
}
```

Vetor

```
#include <stdio.h> //printf (), scanf ().
#include <stdlib.h> //system ().

void preencher (int vet[5]); //Protótipo ou assinatura.
int maior (int vet[5]); //Protótipo ou assinatura.

int main() {
    int max, v[5];

    preencher (v);
    max = maior (v);
    printf ("\n\n\nMaior valor informado: %d \n\n\n", max);
    system ("pause");
    return 0;
}
```

```
void preencher (int vet[5]) {
    int i;

    for (i = 0; i < 5; i++) {
        printf ("\nInforme um numero: ");
        scanf ("%d", &vet[i]);
    }
}

int maior (int vet[5]) {
    int i, m;

    m = vet[0];
    for (i = 1; i < 5; i++) {
        if (vet[i] > m)
            m = vet[i];
    }
    return m;
}
```

Diretiva define

```
#include <stdio.h> //printf (), scanf ().
#include <stdlib.h> //system ().

#define TAM 5

void preencher (int vet[TAM]); //Protótipo ou assinatura.
int maior (int vet[TAM]); //Protótipo ou assinatura.

int main() {
    int max, v[TAM];

    preencher (v);
    max = maior (v);
    printf ("\n\n\nMaior valor informado: %d \n\n\n", max);
    system ("pause");
    return 0;
}
```

```
void preencher (int vet[TAM]) {
    int i;

    for (i = 0; i < TAM; i++) {
        printf ("\nInforme um numero: ");
        scanf ("%d", &vet[i]);
    }
}

int maior (int vet[TAM]) {
    int i, m;

    m = vet[0];
    for (i = 1; i < TAM; i++) {
        if (vet[i] > m)
            m = vet[i];
    }
    return m;
}
```


Matriz

```
#include <stdio.h> //printf ().
#include <stdlib.h> //system (), rand () e srand ().
#include <time.h> //time().
```

```
#define TAM 3
```

```
//Assinaturas ou protótipos das funções/procedimentos.
```

```
void preenche (int mat[TAM][TAM]);
```

```
void imprime (int mat[TAM][TAM]);
```

```
int maior (int mat[TAM][TAM]);
```

```
int main () {
```

```
    int resp, m[TAM][TAM];
```

```
    preenche (m);
```

```
    imprime (m);
```

```
    resp = maior (m);
```

```
    printf ("\nMaior = %d\n\n", resp);
```

```
    system ("pause");
```

```
    return (0);
```

```
}
```

```
void preenche (int m[TAM][TAM]) {
    int i, j;
```

```
    srand (time (NULL)); //Inicia a geração de inteiros aleatórios com uma nova semente.
```

```
    for (i = 0; i < TAM; i++)
```

```
        for (j = 0; j < TAM; j++)
```

```
            m[i][j] = rand () % 100; // Gera valores aleatórios inteiros na faixa de 0 a 99.
```

```
}
```

```
void imprime (int m[TAM][TAM]) {
```

```
    int i, j;
```

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

```
        for (j = 0; j < TAM; j++)
```

```
            printf (" %d ", m[i][j]);
```

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

```
        }
```

```
}
```

```
int maior (int m[TAM][TAM]) {
```

```
    int resp, i, j;
```

```
    resp = m[0][0];
```

```
    for (i = 0; i < TAM; i++)
```

```
        for (j = 0; j < TAM; j++)
```

```
            if (m[i][j] > resp)
```

```
                resp = m[i][j];
```

```
    return resp;
```

```
}
```

Estrutura ou Registro

```
#include <stdio.h> //printf ().
#include <stdlib.h> //system (), rand () e srand ().
#include <time.h> //time().
```

```
struct pessoa {
    int id;
    double salario;
};
```

```
int main () {
    struct pessoa p;

    srand (time (NULL)); //Inicia a geração de inteiros aleatórios com uma nova semente.

    p.id = rand () % 100; //O id é um número aleatório entre 0 e 99.

    printf ("\nInforme o salario: ");
    scanf ("%lf", &(p.salario));

    printf ("\n\n\n ID: %d, Salario: %.2f", p.id, p.salario);
    printf ("\n\n\n");

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

Estrutura ou Registro - typedef

```
#include <stdio.h> //printf ().
#include <stdlib.h> //system (), rand () e srand ().
#include <time.h> //time().
```

```
struct pessoa {
    int id;
    double salario;
};
```

```
typedef struct pessoa tpessoa;
```

```
int main () {
    tpessoa p;

    srand (time (NULL)); //Inicia a geração de inteiros aleatórios com uma nova semente.

    p.id = rand () % 100; //O id é um número aleatório entre 0 e 99.

    printf ("\nInforme o salario: ");
    scanf ("%lf", &(p.salario));

    printf ("\n\n\n ID: %d, Salario: %.2f", p.id, p.salario);
    printf ("\n\n\n");

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

Estrutura ou Registro - typedef

```
#include <stdio.h> //printf ().
#include <stdlib.h> //system (), rand () e srand ().
#include <time.h> //time().
```

```
typedef struct pessoa {
    int id;
    double salario;
} tpessoa;
```

```
int main () {
    tpessoa p;

    srand (time (NULL)); //Inicia a geração de inteiros aleatórios com uma nova semente.

    p.id = rand () % 100; //O id é um número aleatório entre 0 e 99.

    printf ("\nInforme o salario: ");
    scanf ("%lf", &(p.salario));

    printf ("\n\n\n ID: %d, Salario: %.2f", p.id, p.salario);
    printf ("\n\n\n");

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

» una

VEM
PRA
UNA

