



Fundamentos de programação em C

Fundamentos de um ambiente típico de C

Itens necessários para desenvolvimento de programas em C:

- um ambiente de desenvolvimento de programas,
- a linguagem
- a biblioteca padrão de C

Alguns passos típicos para criação de um programa:

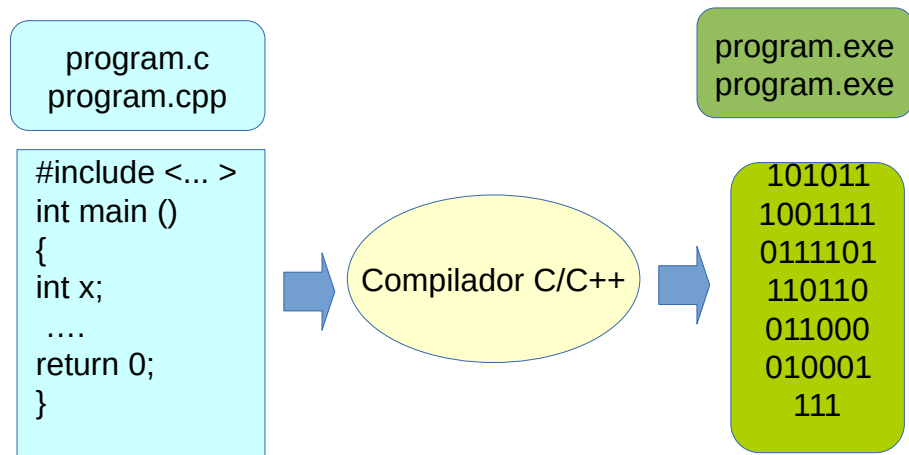
- Criar/editar um arquivo com extensão `.c`
`nome.c`
- Escrever os comandos
- Compilar o programa

Compilação

- O compilador:
 - lê a primeira instrução do programa
 - verifica sintaxe
 - se não houver erro, converte-a para linguagem de máquina
 - segue para a próxima instrução
 - repete o processo até alcançar a última instrução ou achar um erro de sintaxe

Compilação

- Se não houver erros um programa **.obj (.o)** é criado, contendo as instruções já traduzidas.
- Este programa não pode ser executado até que sejam agregadas a ele as funções em linguagem de máquina (das bibliotecas)
- Este trabalho é feito por um programa **linkeditor**, que além de incluir as funções cria um programa executável **.exe**



Estrutura básica de um programa em C

- Um programa em C basicamente é composto por:
 - diretivas de pre processador
 - funções
 - variáveis
 - expressões
 - comentários

Como compilar e executar programa no Linux (Ubuntu)

1. Entrar no terminal (Ctrl+Alt+T)
2. Entrar na pasta do programa
3. Digitar

```
gcc my_source.c -o my_app
```

4. Digitar

```
./my_app
```

Onde

my_source.c – nome do seu arquivo .c

my_app - nome do arquivo executável a ser criado

Dicas de instalação

- Software usado Code::Blocks IDE

<http://www.codeblocks.org/downloads/26>

- Instalação no Ubuntu
 - Instalar pelo Ubuntu Software
 - Ou seguir as dicas da pagina

<https://www.linuxbabe.com/ubuntu/install-code-blocks-ubuntu-16-04-17-04>

- Se o Code::Blocks não aparecer na lista reiniciar o computador
- Tomar cuidado com a configuração do compilador

Code::Blocks IDE

- Home
- Features
- Screenshots
- Downloads
 - Binaries
 - Source
 - SVN
- Plugins
- User manual
- Licensing
- Donations

Quick links

- FAQ
- Wiki
- Forums
- Forums (mobile)
- Nightlies
- Ticket System
- Browse SVN
- Browse SVN log



- Windows XP / Vista / 7 / 8.x / 10
- Linux 32 and 64-bit
- Mac OS X

NOTE: For older OS'es use older releases. There are releases for many OS version and platforms on the **Sourceforge.net** page.

NOTE: There are also more recent *nightly builds* available in the **forums** or (for Debian and Fedora users) in **Jens' Debian repository** and **Jens' Fedora repository**. Please note that we consider nightly builds to be *stable*, usually.

NOTE: We have a **Changelog for 17.12**, that gives you an overview over the enhancements and fixes we have put in the new release.



Windows XP / Vista / 7 / 8.x / 10:

File	Date	Download from
codeblocks-17.12-setup.exe	30 Dec 2017	FossHUB or Sourceforge.net
codeblocks-17.12-setup-nonadmin.exe	30 Dec 2017	FossHUB or Sourceforge.net
codeblocks-17.12-nosetup.zip	30 Dec 2017	FossHUB or Sourceforge.net
codeblocks-17.12mingw-setup.exe	30 Dec 2017	FossHUB or Sourceforge.net
codeblocks-17.12mingw-nosetup.zip	30 Dec 2017	FossHUB or Sourceforge.net
codeblocks-17.12mingw_fortran-setup.exe	30 Dec 2017	FossHUB or Sourceforge.net

NOTE: The codeblocks-17.12-setup.exe file includes Code::Blocks with all plugins. The codeblocks-17.12-setup-nonadmin.exe file is provided for convenience to users that do not have administrator rights on their machine(s).

NOTE: The codeblocks-17.12mingw-setup.exe file includes *additionally* the GCC/G++ compiler and GDB debugger from **TDM-GCC** (version 5.1.0, 32 bit, SJLJ). The codeblocks-17.12mingw_fortran-setup.exe file includes *additionally to that* the GFortran compiler (**TDM-GCC**).

NOTE: The codeblocks-17.12/mingw/-nosetup.zip files are provided for convenience to users that are allergic against installers. However, it will not allow to select

Edição e compilação online

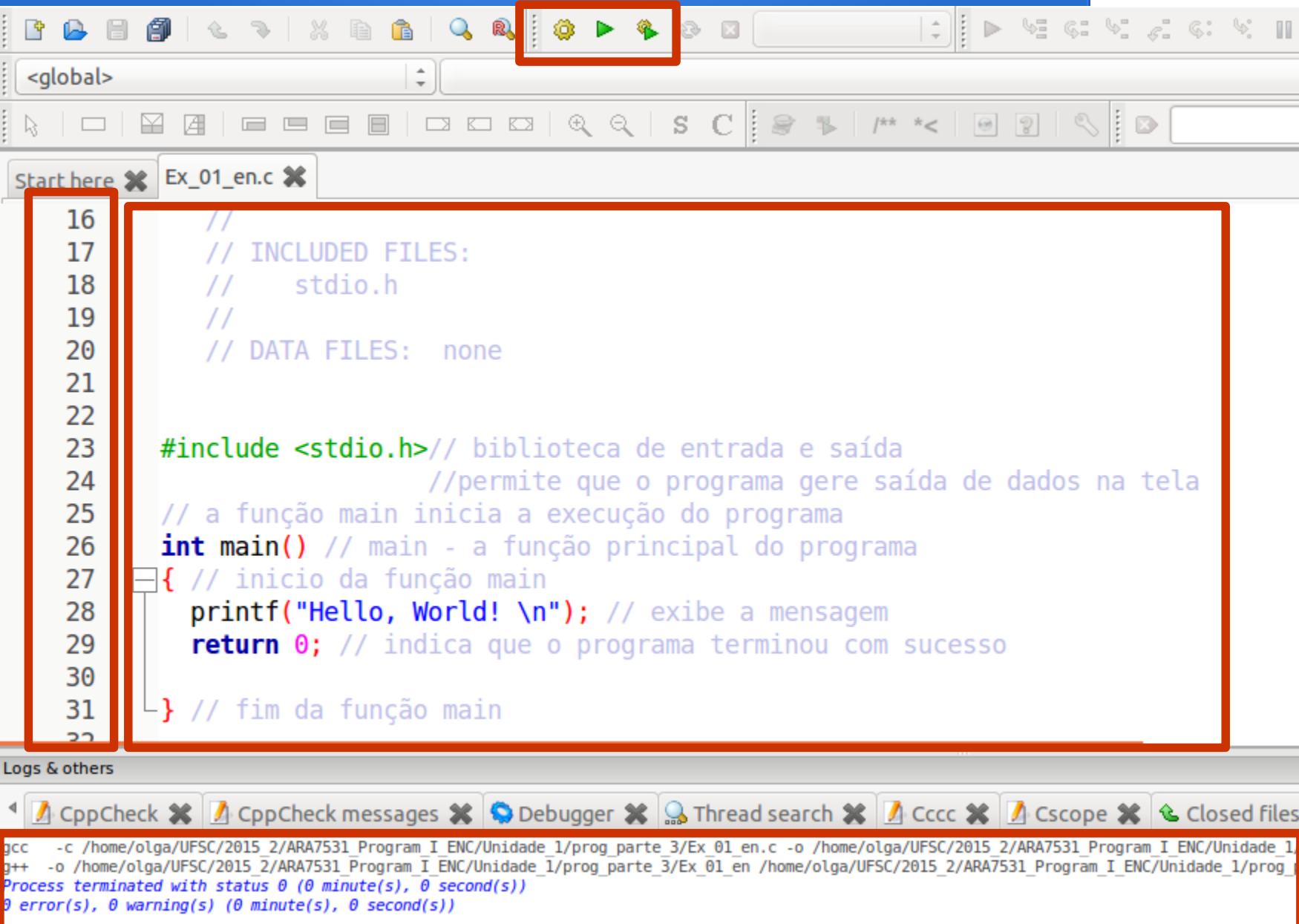
- Editor de código e compilador online

OnlineGDB - online compiler and debugger for c/c++

<https://www.onlinegdb.com/>

The screenshot displays the OnlineGDB web interface. On the left is a blue sidebar with navigation links: 'Welcome, Eva', 'Create New Project', 'My Projects', 'Classroom new', 'Learn Programming', 'Programming Questions', 'Logout', and footer links like 'About', 'FAQ', 'Blog', 'Terms of Use', 'Contact Us', 'GDB Tutorial', 'Credits', and 'Privacy'. The main area features a code editor with a file named 'main.c' containing a C program that prints 'Hello World'. Above the editor is a toolbar with buttons for 'Run' (highlighted with a red box), 'Debug', 'Stop', 'Share', 'Save' (highlighted with a green box), 'Beautify', and a 'Language' dropdown menu set to 'C' (highlighted with an orange box). The right sidebar shows debugging tools: 'Call Stack', 'Local Variables', 'Registers', 'Display Expressions', and 'Breakpoints and Watchpoints'. At the bottom, there is an 'input' section with a 'Command line' and a 'Standard Input' selector with radio buttons for 'Interactive Console' (selected) and 'Text'.

Exemplo 1: mensagem de texto



The screenshot shows a C++ IDE with a file named `Ex_01_en.c`. The code is a simple program that prints "Hello, World!". The IDE has a toolbar at the top with icons for file operations, editing, and running. The code editor shows the following code:

```
16 //
17 // INCLUDED FILES:
18 //     stdio.h
19 //
20 // DATA FILES: none
21
22
23 #include <stdio.h> // biblioteca de entrada e saída
24 // permite que o programa gere saída de dados na tela
25 // a função main inicia a execução do programa
26 int main() // main - a função principal do programa
27 { // inicio da função main
28     printf("Hello, World! \n"); // exibe a mensagem
29     return 0; // indica que o programa terminou com sucesso
30 }
31 // fim da função main
32
```

The IDE also shows a "Logs & others" panel at the bottom with the following output:

```
gcc -c /home/olga/UFSC/2015_2/ARA7531_Program_I_ENC/Unidade_1/prog_parte_3/Ex_01_en.c -o /home/olga/UFSC/2015_2/ARA7531_Program_I_ENC/Unidade_1/prog_parte_3/Ex_01_en.o
g++ -o /home/olga/UFSC/2015_2/ARA7531_Program_I_ENC/Unidade_1/prog_parte_3/Ex_01_en /home/olga/UFSC/2015_2/ARA7531_Program_I_ENC/Unidade_1/prog_parte_3/Ex_01_en.o
Process terminated with status 0 (0 minute(s), 0 second(s))
0 error(s), 0 warning(s) (0 minute(s), 0 second(s))
```

Estrutura básica de um programa em C

<code>int main ()</code>	a função que inicia a execução do programa
tipo <code>int</code>	significa que a função deverá retornar um número inteiro
<code>return 0;</code>	instrução de retorno
<code>()</code>	parênteses após o nome <code>main ()</code> indicam para o compilador que se trata de uma função e não se refere a uma variável
<code>{ }</code>	toda função deve começar com uma chave de abertura de bloco <code>{</code> e terminar com uma chave de fechamento de bloco <code>}</code> As chaves delimitam o corpo da função
<code>#include <Nome da biblioteca 1></code>	a diretiva de pré-processador (não é a linguagem C) Provoca a inclusão de outro arquivo em nosso programa O compilador substitui essa linha pelo conteúdo do arquivos indicado antes do programa ser compilado

Comentários / Documentação

//Meu 1o programa em C

- **//** indica que o restante de cada linha é um **comentário**.
- Os comentários
 - servem para documentar programas e melhorar a legibilidade dos mesmos.
 - ajudam as outras pessoas a ler e entender seu programa.
 - não levam o computador a executar qualquer ação quando o programa for executado.
 - são ignorados pelo compilador e não causam qualquer geração de código objeto em linguagem de máquina.
 - O comentário inicial geralmente descreve o propósito do programa.
 - Um comentário que começa com **//** é chamado de comentário de linha única porque o comentário termina no fim da linha em que está.
 - comentário de linhas múltiplas - começa com **/*** e termina com ***/**

Padrão de documentação

- Um programa bem escrito deve ser documentado
- Objetivo da documentação é facilitar a manutenção do código e possibilitar a troca de membros de equipe de desenvolvimento.
- Documentação do código segue um certo padrão
- Um dos exemplos de como deve ser feita a documentação encontra-se nesse link:

<https://www.bgsu.edu/arts-and-sciences/computer-science/cs-documentation/c-plus-plus-program-documentation-guidelines.html>

Exemplo 1: Documentação

```
// FILE:      Ex_01_en.c
// TITLE:     My first program in C
// SUBMITTED BY: Student Name
//           All code is my own except where credited to others.
// FOR COURSE: Programming I
// DUE DATE:  August 21, 2015
//
// PURPOSE:
// This program will print one text line "Hello, World!"
//
// OVERALL METHOD:
// The list of general tasks is:
// 1. Print text
//
// FUNCTIONS: none
//
// INCLUDED FILES:
//  stdio.h
//
// DATA FILES: none
```

Inclusão de bibliotecas

- `#include <stdio.h>`
- Solicita ao compilador que inclua o arquivo `stdio.h` em nosso programa fonte antes de compilá-lo.
- O arquivo `stdio.h` (standard input/output) contém as definições e declarações necessárias para entrada/saída de dados.
- No nosso caso a função `printf()`.

Função principal

`int main()`

- faz parte de todo programa em C/C++.
- Os parênteses depois de `main` indicam que `main` é um bloco de construção de programa chamado de `função`.
- Os programas contêm uma ou mais funções, exatamente uma das quais deve ser `main`.
- Os programas começam a execução na função `main`, ainda que `main` não seja a primeira função do programa.
- A palavra-chave `int`, à esquerda de `main` indica que `main` devolve um inteiro (número inteiro).
- A chave à esquerda, `{`, deve começar o corpo de qualquer função.
- Uma chave à direita, `}`, correspondente deve terminar o corpo de cada função.

Saída de dados: função `printf()`

`printf()`

- é uma das funções de I/O (input/output – entrada e saída) da biblioteca padrão fornecida com os compiladores C.
- está associada à saída padrão do sistema operacional.
- a informação dentro dos parênteses (“`Hello, World!`”) é chamada de argumento – é algo que vai ser exibido na tela.

;

Toda instrução em C deve terminar com um ponto e vírgula ;

Erro comum de programação

- Omitir o ponto e vírgula no fim de um comando é um erro de sintaxe.
- Um erro de sintaxe ocorre quando o compilador não consegue reconhecer um comando.
- O compilador normalmente emite uma mensagem de erro para ajudar o programador a localizar e corrigir o comando incorreto.
- Os erros de sintaxe são também chamados de **erros de compilação** porque aparecem na fase de compilação do programa.

return 0;

- indica que o programa terminou com sucesso
- é incluída no fim de toda função **main**.
- A palavra-chave **return** de C/C++ é um dos vários meios para sair de uma função.
- Quando o comando **return** for usado no fim de **main**, o valor **0** indica que o programa terminou com sucesso.
- As chaves **{ }** indicam o início e o final de função **main**

Espaços em branco e tabulações

- Podemos inserir espaços em branco, tabulações e pular as linhas
- O compilador ignora esses caracteres
- Podemos escrever várias instruções em uma única linha, separados por espaços/tabulações
- Podemos escrever uma instrução em varias linhas
- Não podemos inserir espaços em branco no meio de identificadores ou símbolos da linguagem

Boa prática de programação

- Escreva seus programas de uma maneira simples e direta.
- Isto é às vezes chamado de **KIS** (“mantenha-o simples” - Keep It Simple).
- Não “force” a linguagem tentando usos estranhos.

Variáveis

- São lugares (posições) na memória que servem para armazenar dados.
- As variáveis são acessadas através de um **identificador** único.
- O conteúdo de uma variável pode variar ao longo do tempo durante a execução de um programa.
- Uma variável só pode armazenar um valor a cada instante.

Identificador

- Um identificador é o nome usado para identificar uma variável, função ou algum outro elemento definido pelo programador.
- As regras básicas para a formação dos identificadores são:
 - Os caracteres que podem ser utilizados são: os números, as letras maiúsculas, as letras minúsculas e o caractere sublinhado
 - O primeiro caractere deve ser sempre uma letra ou o caractere sublinhado
 - Não são permitidos espaços em branco e caracteres especiais (@, \$, +, -, %, !)
 - Não se pode usar as palavras reservadas nos identificadores, ou seja, palavras que pertençam a uma linguagem de programação

Identificação de Variáveis

- Exemplos (**corretos**):
 - A1
 - X
 - sum
 - t_total
- Exemplos (**incorretos**):
 - 1A
 - Preço
 - @email
 - Codigo#curso

Palavras reservadas

C/C++

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

C++

asm	bool	catch	class	delete
false	friend	new	private	protected
public	template	this	throw	true

Tipos de Variáveis

- Um tipo classifica o dado que uma determinada variável pode receber.
- As variáveis só podem armazenar valores do mesmo tipo em que foram definidas.
- De forma geral uma variável pode ser de um dos seguintes tipos:
 - **Lógico:** armazena valores do tipo verdadeiro ou falso.
 - **Numérico:** armazena números inteiros ou reais.
 - Pode ser **inteiro** e **real**
 - **Literal:** armazena um único carácter ou uma sequência de caracteres.
 - Declarado como **caracter**.

Tipos de dados numéricos inteiros

Tipo	Tamanho	Valores
char	1 byte	-128 até 127; ou 0 até 255
unsigned char	1 byte	0 até 255
signed char	1 byte	-128 até 127
int	2 ou 4 bytes	-32 768 até 32 767; ou -2 147 483 648 até 2 147 483 647
unsigned int	2 ou 4 bytes	0 até 65 535; ou 0 até 4 294 967 295
short	2 bytes	-32 768 até 32 767
unsigned short	2 bytes	0 até 65 535
long	4 bytes	-2 147 483 648 até 2 147 483 647
unsigned long	4 bytes	0 até 4 294 967 295

Tipos de dados numéricos com ponto flutuante

Tipo	Tamanho	Valores	Precisão
float	4 byte	1.2E-38 até 3.4E+38	6 posições decimais
double	8 byte	2.3E-308 até 1.7E+308	15 posições decimais
long double	10 byte	3.4E-4932 até 1.1E+4932	19 posições decimais

Declaração de Variáveis

- As variáveis são armazenadas em memória.
- Antes de serem usadas as variáveis precisam ser **declaradas**.
- A declaração consiste em definir um **identificador** e um **tipo** para a variável, no seguinte formato:

tipo identificador;

Exemplo:

int x;

float y;

Operações matemáticas

Os operadores aritméticos são todos operadores binários, isto é, operadores que recebem dois operandos.

Operação	Operador aritmético	Expressão algébrica	Expressão em C
Adição	+	$f + 7$	$f + 7$
Subtração	-	$p - c$	$p - c$
Multiplicação	*	bm	$b*m$
Divisão	/	x/y ou $x \div y$	x/y
Resto de divisão	%	$r \bmod s$	$r \% s$

Divisão “inteira”

- A divisão “inteira” / (isto é, tanto o numerador como o denominador são inteiros) gera como resultado um número inteiro
- Por exemplo, a expressão $7 / 4$ fornece o valor 1, e a expressão $17 / 5$ fornece o valor 3.
- Note que qualquer parte fracionária na divisão inteira é simplesmente descartada (isto é, truncada) - não ocorre nenhum arredondamento.

Operador % (resto de divisão)

- C oferece o operador “módulo” - %, que retorna o resto da divisão inteira.
- O operador módulo pode ser usado somente com operandos inteiros.
- A expressão $x \% y$ fornece o resto depois de x ser dividido por y .
- Assim,

$$7 \% 4 = 3$$

$$17 \% 5 = 2$$

Ordem de precedência

- Os parênteses são usados em expressões quase do mesmo modo que nas expressões algébricas.
- Por exemplo, para multiplicar a pela quantidade $b + c$ escrevemos:

$$a * (b + c)$$

Ordem de precedência

C aplica os operadores em expressões aritméticas em uma sequência precisa, determinada pelas regras de prioridade de operadores, que em geral são as mesmas que aquelas seguidas na álgebra:

- 1) Operadores em expressões contidas dentro de pares de parênteses são calculados primeiro.
 - Assim, os parênteses podem ser usados para forçar que a ordem de cálculo aconteça em qualquer sequência desejada pelo programador.
 - Dizemos que os parênteses estão no “nível mais alto de prioridade”.
 - Em casos de parênteses aninhados, ou embutidos, os operadores no par mais interno de parênteses são aplicados primeiro.

Ordem de precedência

- 2) As operações de multiplicação, divisão e módulo são aplicadas em seguida.
 - Se uma expressão contém várias operações de multiplicação, divisão e módulo, os operadores são aplicados da esquerda para a direita.
 - Dizemos que a multiplicação, a divisão e o módulo estão no mesmo nível de prioridade.
- 3) As operações de adição e de subtração são aplicadas por último. Se uma expressão contém várias operações de adição e subtração, os operadores são aplicados da esquerda para a direita.
 - A adição e subtração têm, também, o mesmo nível de prioridade.

Entrada/Saída de dados: caracteres/códigos especiais

- Além do caractere ENTER, outros caracteres não podem ser digitados diretamente do teclado para dentro do programa ou exibidos na tela.
- Esses caracteres são codificados como combinação do sinal \ com outros caracteres.
- Alguns exemplos:

\n	Nova linha. Posiciona o cursor da tela no início da próxima linha.
\t	Tabulação. Move o cursor da tela para a próxima posição de tabulação.
\r	Posiciona o cursor da tela no início da linha atual. (Não avança para a próxima linha)
\\	Barra invertida. Usado para caractere \
\”	Aspas duplas
\'	Aspas simples

Códigos de formatação para printf()

%d	inteiro decimal com sinal	int
%i	inteiro decimal com sinal	int
%f	ponto flutuante (número real) (single precision floating-point value)	float
%lf	ponto flutuante (número real) de dupla precisão (long precision floating-point value - double)	double
%c	um único caractere	char
%s	uma sequencia de caracteres	string
%p	ponteiro	

Entrada de dados: função **scanf()**

- complemento de **printf()**
- é uma função presente na biblioteca padrão
- permite ler dados da entrada padrão
- os argumentos de **scanf()** devem ser endereços de variáveis

Formas de uso:

scanf("%d", &x);

- lê um numero inteiro que tenha sido digitado no teclado
- o valor lido será copiado para a variável **x**

scanf("%c", &s);

- lê um caractere que tenha sido digitado no teclado.
- o valor lido será copiado para a variável **s**

scanf("%d %c %f ", &x, &s, &taxa);

- os valores lidos serão copiados para as variáveis, **x**, **s** e **taxa**
- lê um número inteiro, um caractere e um número real.

Erros de execução

Erro comum de programação

- Erros como os de divisão por zero acontecem quando um programa está sendo executado.
- Por isso, estes erros são chamados de **erros durante a execução**.
- Dividir por zero é geralmente um erro fatal, isto é, um erro que causa o término imediato do programa sem este ter executado seu trabalho com sucesso.
- Os erros não-fatais permitem que os programas sejam executados até a conclusão, frequentemente produzindo resultados incorretos. (Nota: em alguns sistemas, dividir por zero não é um erro fatal.)

Símbolos utilizados no Fluxograma



Símbolo utilizado para indicar o início e o fim do algoritmo.



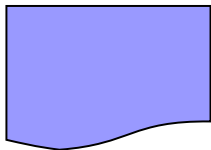
Permite indicar o sentido do fluxo de dados. Serve exclusivamente para conectar os símbolos ou blocos existentes.



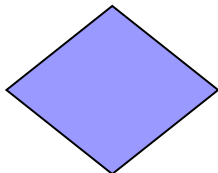
Símbolo utilizado para indicar cálculos e atribuições de valores.



Símbolo utilizado para representar a entrada de dados.



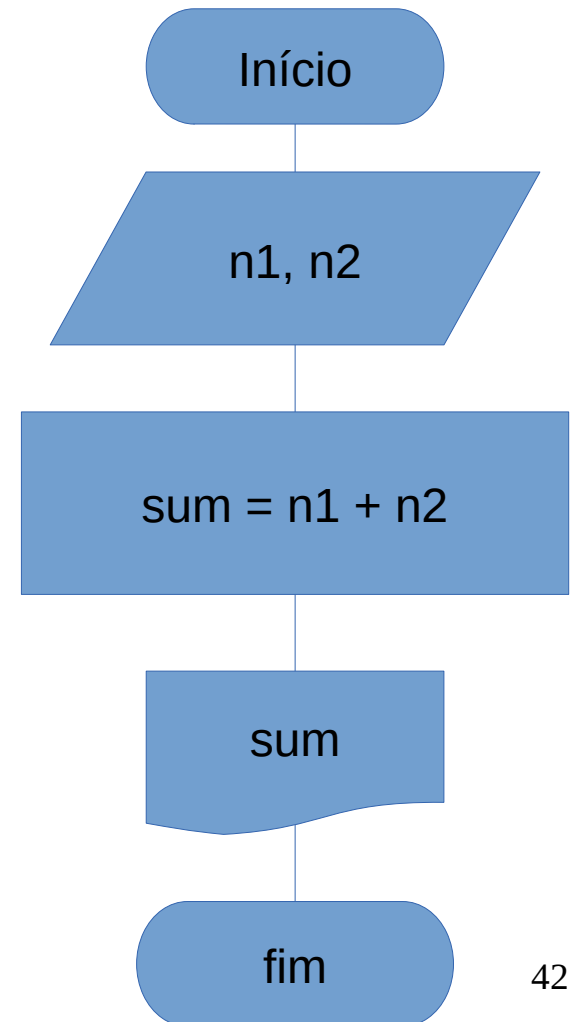
Símbolo utilizado para representar a saída de dados..



Símbolo utilizado para indicar que deve ser tomada uma decisão, apontando a possibilidade de desvios.

Exemplo 2: soma de dois números inteiros

- Calcule a soma de dois números inteiros fornecidos pelo usuário.
- Algoritmo:
 - **Passo 1:** Receber dois números que serão somados.
 - **Passo 2:** Somar os números
 - **Passo 3:** Mostrar o resultado obtido na soma.

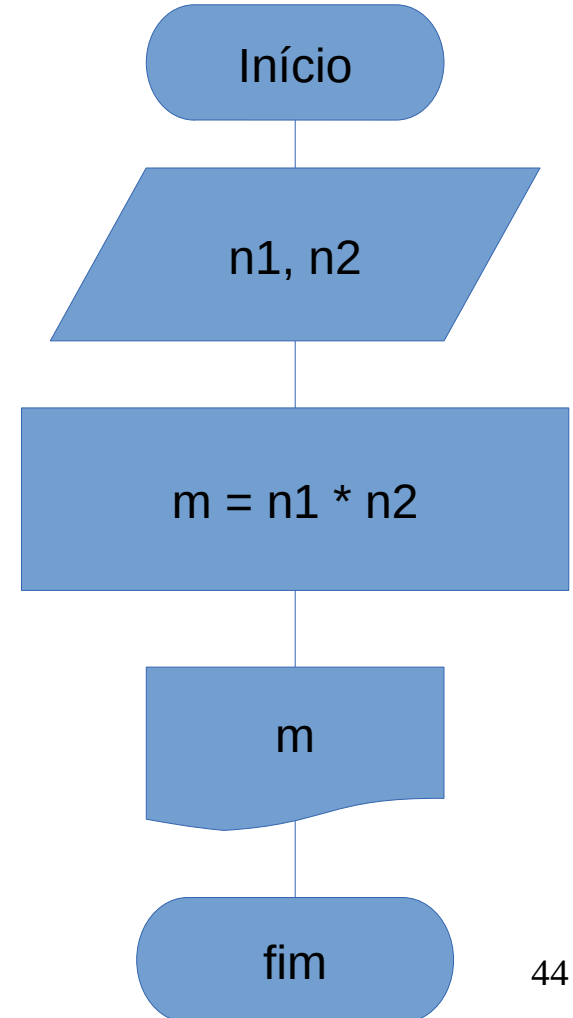


Exemplo 2: soma de dois números inteiros

```
1 //Programa calcula a soma de 2 numeros inteiros
2 #include <stdio.h>
3
4 int main()
5 {
6     // declaração de variaveis
7     int n1, n2; // numeros a serem recebidos
8     int sum;    // resultado
9
10    printf("Soma de dois números inteiros: \n ");
11    // Entrada de dados: recebe 2 numeros
12    printf("Informe o 1o numero: ");
13    scanf("%i", &n1);
14
15    printf("Informe o 2o numero: ");
16    scanf("%i", &n2);
17
18    // Processamento: soma dos numeros digitados
19    sum = n1 + n2;
20
21    // Saída de dados: Mostra o resultado da soma
22    printf("\n Resultado do programa: ");
23    printf("\n Soma = %i \n", sum);
24
25    return 0;
26 }
```

Exemplo 3: multiplicação de dois números inteiros

- Calcule o resultado da multiplicação de dois números inteiros fornecidos pelo usuário.
- Algoritmo:
 - Passo 1: Receber dois números que serão multiplicados.
 - Passo 2: Multiplicar os números
 - Passo 3: Mostrar o resultado obtido na multiplicação.

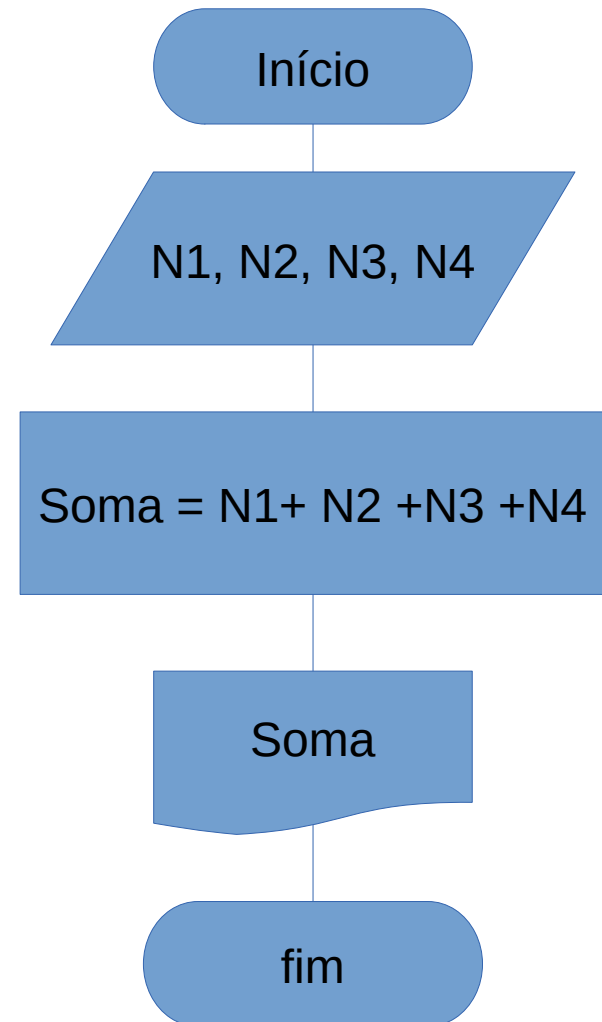


Exemplo 3: multiplicação de dois números inteiros

```
1 //Calcula o resultado da multiplicação de 2 numeros inteiros
2 #include <stdio.h>
3
4 int main()
5 {
6     // declaração de variaveis
7     int n1, n2; // numeros a serem recebidos
8     int m;      // resultado
9
10    printf("Multiplicação de dois números inteiros: \n ");
11    // Entrada de dados: recebe 2 numeros
12    printf("Informe o 1o numero: ");
13    scanf("%i", &n1);
14
15    printf("Informe o 2o numero: ");
16    scanf("%i", &n2);
17
18    // Processamento: multiplicação dos numeros digitados
19    m = n1 * n2;
20
21    // Saída de dados: Mostra o resultado da multiplicação
22    printf("\n Resultado do programa: ");
23    printf("\n m = %i \n", m);
24
25    return 0;
26 }
```

Exemplo 4: soma de quatro números inteiros

- Faça um programa que receba quatro números inteiros, calcule e mostre a soma desses números.

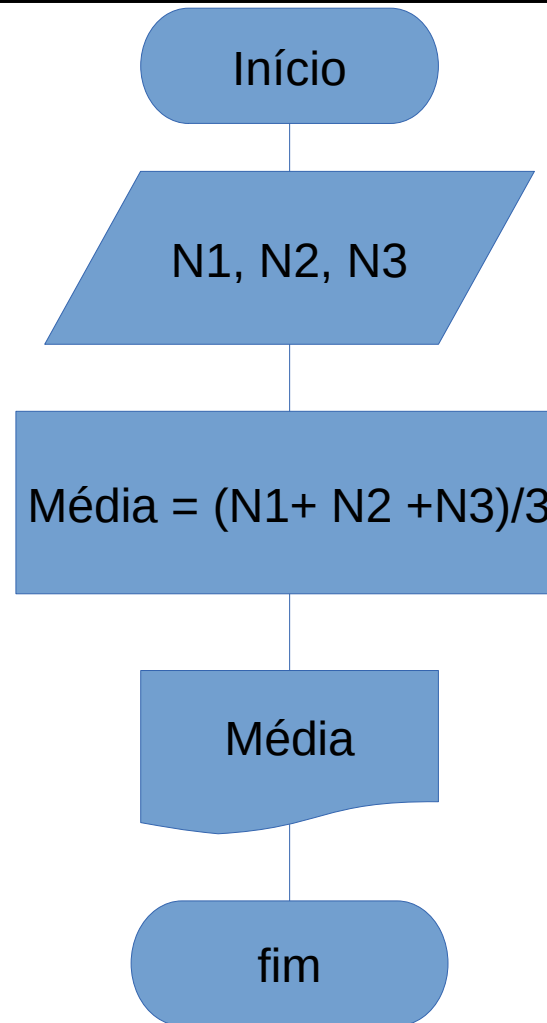


Exemplo 4: soma de quatro números inteiros

```
4  int main()
5  {
6      // declaração de variaveis
7      int n1, n2, n3, n4; // numeros a serem recebidos
8      int sum;           // resultado
9
10     printf("Soma de quatro números inteiros: \n ");
11     // Entrada de dados: recebe 4 numeros
12     printf("Informe o 1o numero: ");
13     scanf("%i", &n1);
14
15     printf("Informe o 2o numero: ");
16     scanf("%i", &n2);
17
18     printf("Informe o 3o numero: ");
19     scanf("%i", &n3);
20
21     printf("Informe o 4o numero: ");
22     scanf("%i", &n4);
23
24     // Processamento: soma dos numeros digitados
25     sum = n1 + n2 + n3 + n4;
26
27     // Saída de dados: Mostra o resultado da soma
28     printf("\n Resultado do programa: ");
29     printf("\n Soma = %i \n", sum);
30
31     return 0;
32 }
```

Exemplo 5: média aritmética

- Faça um programa que receba três notas, calcule e mostre a média aritmética entre elas.
- Obs.: vamos usar números reais

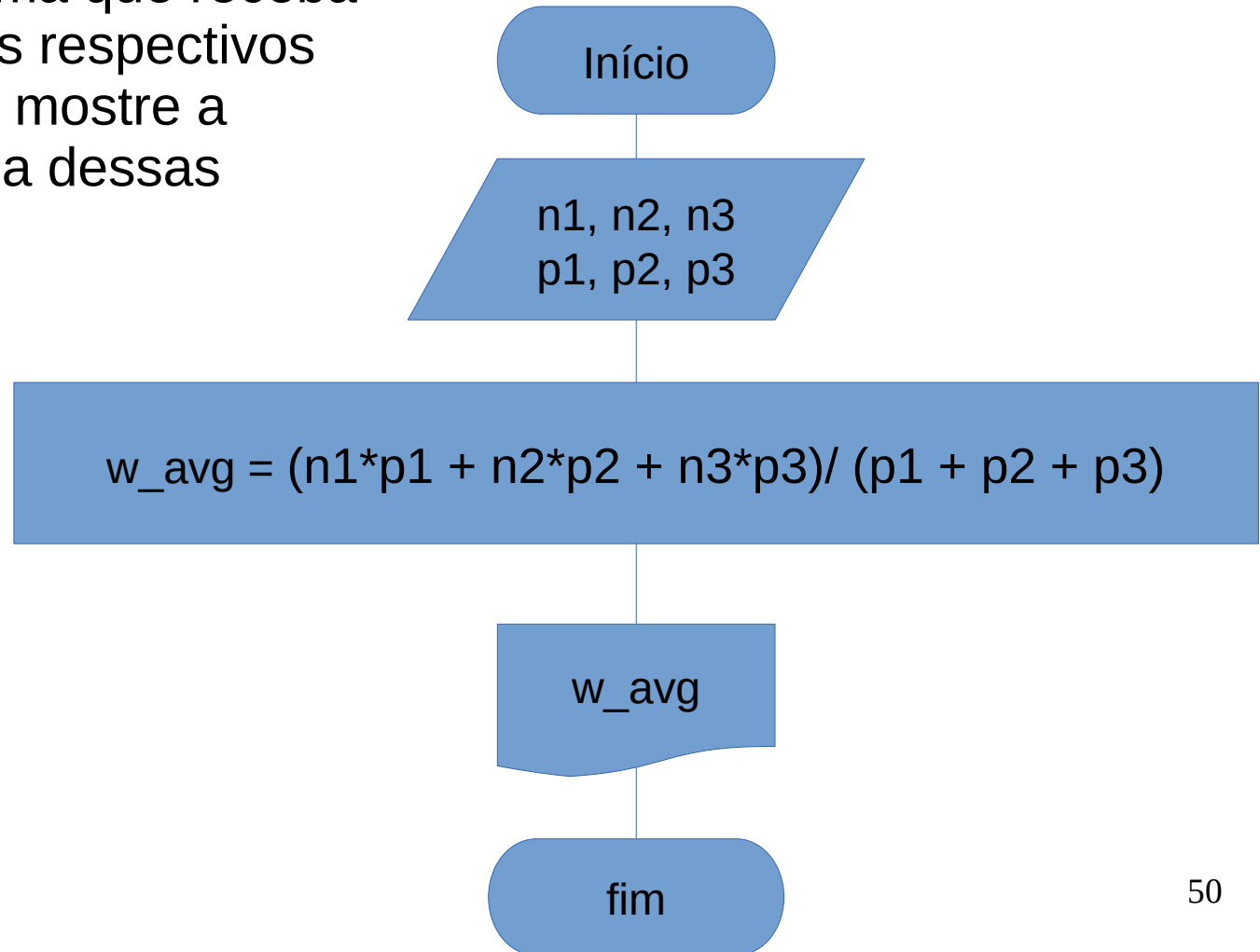


Exemplo 5: média aritmética

```
4  int main()
5  {
6      // declaração de variaveis
7      float n1, n2, n3; // notas
8      float mean;       // média
9
10     printf("Média de três notas: \n ");
11     // Entrada de dados: três notas
12     printf("Informe a 1a nota: ");
13     scanf("%f",&n1);
14
15     printf("Informe a 2a nota: ");
16     scanf("%f",&n2);
17
18     printf("Informe a 3a nota: ");
19     scanf("%f",&n3);
20
21     // Processamento: Calcula a media
22     mean = (n1 + n2 + n3) / 3;
23
24     // Saída de dados: Mostra o resultado da media
25     printf("\n Resultado do programa: ");
26     // %.2f - duas posições depois do ponto decimal: ____ .XX
27     printf("\n Média = %.2f \n", mean);
28
29     return 0;
30 }
```

Exemplo 6: média ponderada

- Faça um programa que receba três notas e seus respectivos pesos, calcule e mostre a média ponderada dessas notas.



Exemplo 6: média ponderada

```
6 // declaração de variaveis
7 float n1, n2, n3; // notas
8 float p1, p2, p3; // pesos
9 float w_avg;      // média ponderada (Weighted average)
10
11 printf("Média ponderada de três notas: \n\n ");
12 // Entrada de dados: as notas e os pesos
13 printf("Informe a 1a nota: ");
14 scanf("%f",&n1);
15
16 printf("Informe o peso da 1a nota: ");
17 scanf("%f",&p1);
18
19 printf("\n Informe a 2a nota: ");
20 scanf("%f",&n2);
21
22 printf("Informe o peso da 2a nota: ");
23 scanf("%f",&p2);
24
25 printf("\n Informe a 3a nota: ");
26 scanf("%f",&n3);
27
28 printf("Informe o peso da 3a nota: ");
29 scanf("%f",&p3);
30
31 // Processamento: Calcula a media ponderada
32 w_avg = (n1*p1 + n2*p2 + n3*p3) / (p1 + p2 + p3) ;
33
34 // Saída de dados: Mostra o resultado da media ponderada
35 printf("\n Resultado do programa: ");
36 // %.2f - duas posições depois do ponto decimal: ____XX
37 printf("\n Média ponderada = %.2f \n", w_avg);
38
```