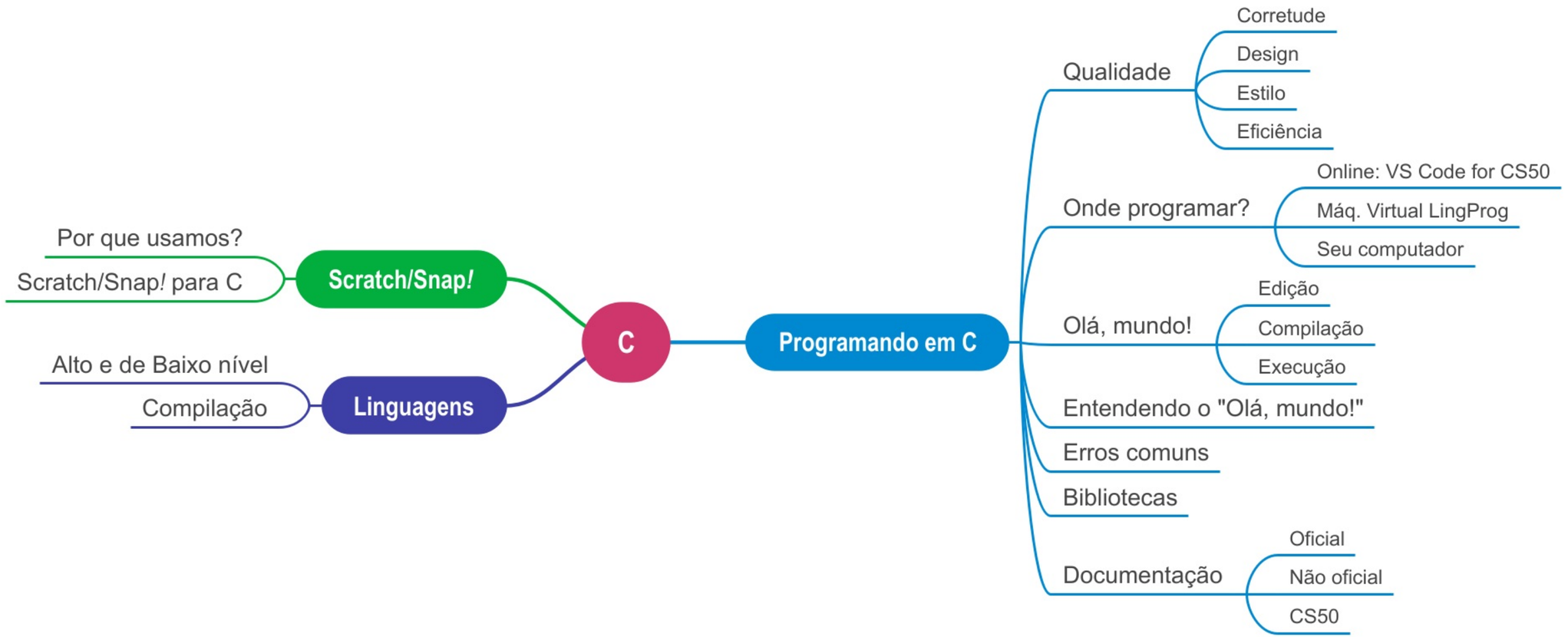
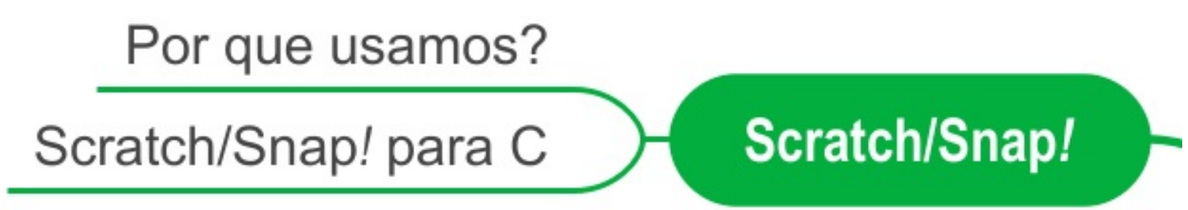
LINGUAGEM C


complexidade
de algoritmos
...
arrays
...
ponteiros
...
compilação
...
estruturas
de dados
internet
algoritmos
fundamentos da programação
pensamento
computacional
memória
C
web
linux
fundamentos da computação

Linguagem C


C
Scratch/Snap!
Por que usamos?
Scratch/Snap! para C
Linguagens
Alto e de Baixo nível
Compilação
Programando em C
Qualidade
Corretude
Design
Estilo
Eficiência
Onde programar?
Online: VS Code for CS50
Máq. Virtual LingProg
Seu computador
Olá, mundo!
Edição
Compilação
Execução
Entendendo o "Olá, mundo!"
Erros comuns
Bibliotecas
Documentação
Oficial
Não oficial
CS50

Lembrete: por que usamos Scratch e Snap*!* ?



Lembrete: por que usamos Scratch e Snap*!* ?

Aprendemos a programar!

- Operações sequenciais
- Operações de repetição
- Operações de decisão e exp. booleanas
- Estruturas de dados (variáveis, arrays)

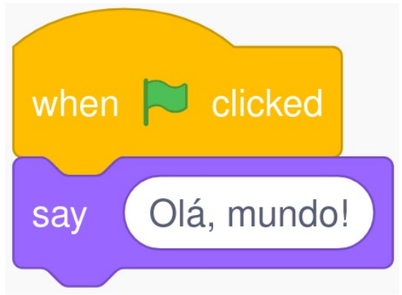
Aprendemos pensamento computacional!

- Decomposição, padrões, abstração, algoritmos, estruturas de dados

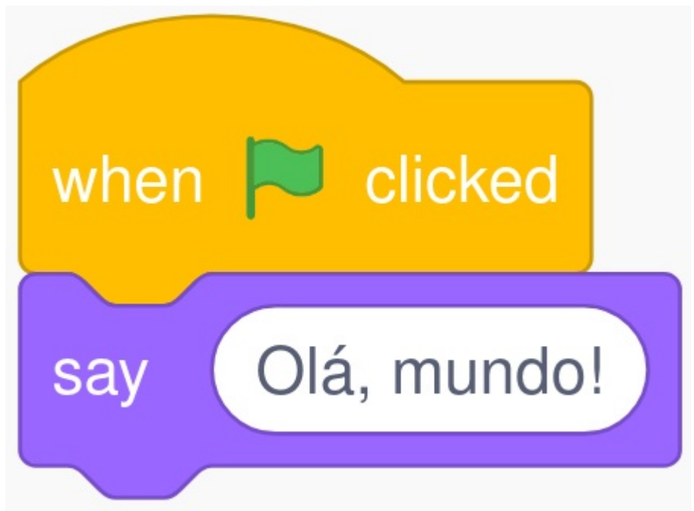
Aprendemos sobre subprogramas:

- Funções, predicados e procedimentos
- Parâmetros e argumentos
- Retorno e efeito colateral
- Funções puras

Aprendemos conceitos avançados diversos.



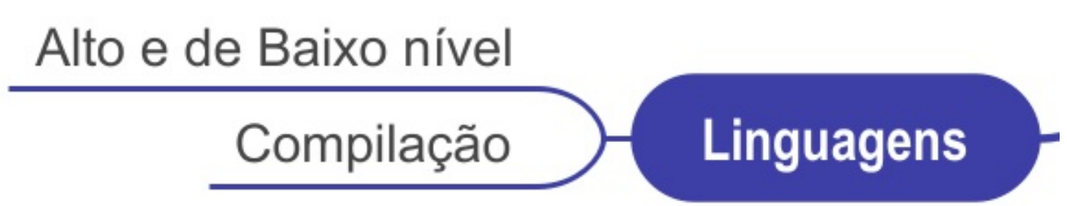
Transição: Scratch/Snap! para C




```c
#include <stdio.h>

int main(void)
{
    printf("Olá, mundo!\n");
}
```

Linguagens de Alto Nível x Linguagens de Baixo Nível



Linguagens de Alto Nível x Linguagens de Baixo Nível

Linguagens de Alto Nível

O programador escreve os algoritmos em uma linguagem de fácil entendimento, semelhante a um texto (inglês).

O programador salva o programa em um arquivo chamado de código fonte.

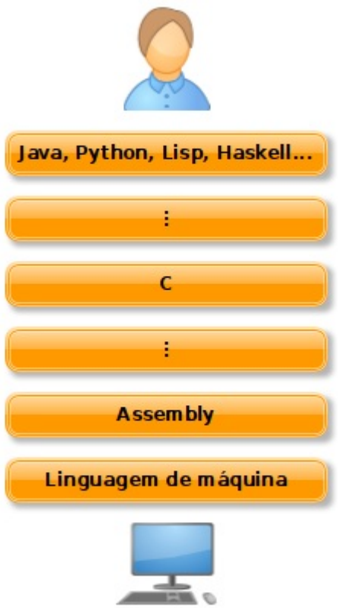
Não importa a linguagem (C, Scratch, Python, etc), todo programa deve ser escrito e salvo em um arquivo de código fonte.

```c
#include <stdio.h>

int main(void)
{
    printf("Olá, mundo!\n");
}
```

Linguagens de Baixo Nível

O programador escreve os algoritmos em uma linguagem semelhante ao que, de fato, o computador entende. A linguagem mais baixo nível que existe é a linguagem de máquina (também chamada de código de máquina).



A linguagem de máquina corresponde aos bits (0 e 1) do seu programa. Contém dados e instruções.

```
01111111 01000101 01001100 01000110 00000010
00000000 00000000 00000000 00000000 00000000
00000010 00000000 00111110 00000000 00000001
10110000 00000101 01000000 00000000 00000000
01000000 00000000 00000000 00000000 00000000
11010000 00010011 00000000 00000000 00000000
00000000 00000000 00000000 00000000 01000000
00001001 00000000 01000000 00000000 00100100
```

Como transformar o código fonte em código de máquina?

Se o computador só entende linguagem de máquina (bits 0 e 1) mas o programador escreve em uma linguagem de alto nível, como fazer a transformação entre a linguagem de alto nível para a linguagem de máquina?

Podemos encarar essa transformação com o mesmo modelo geral da computação:

Como transformar o código fonte em código de máquina?

```c
#include <stdio.h>

int main(void)
{
    printf("Olá, mundo!\n");
}
```

input
(código fonte)

COMPILADOR

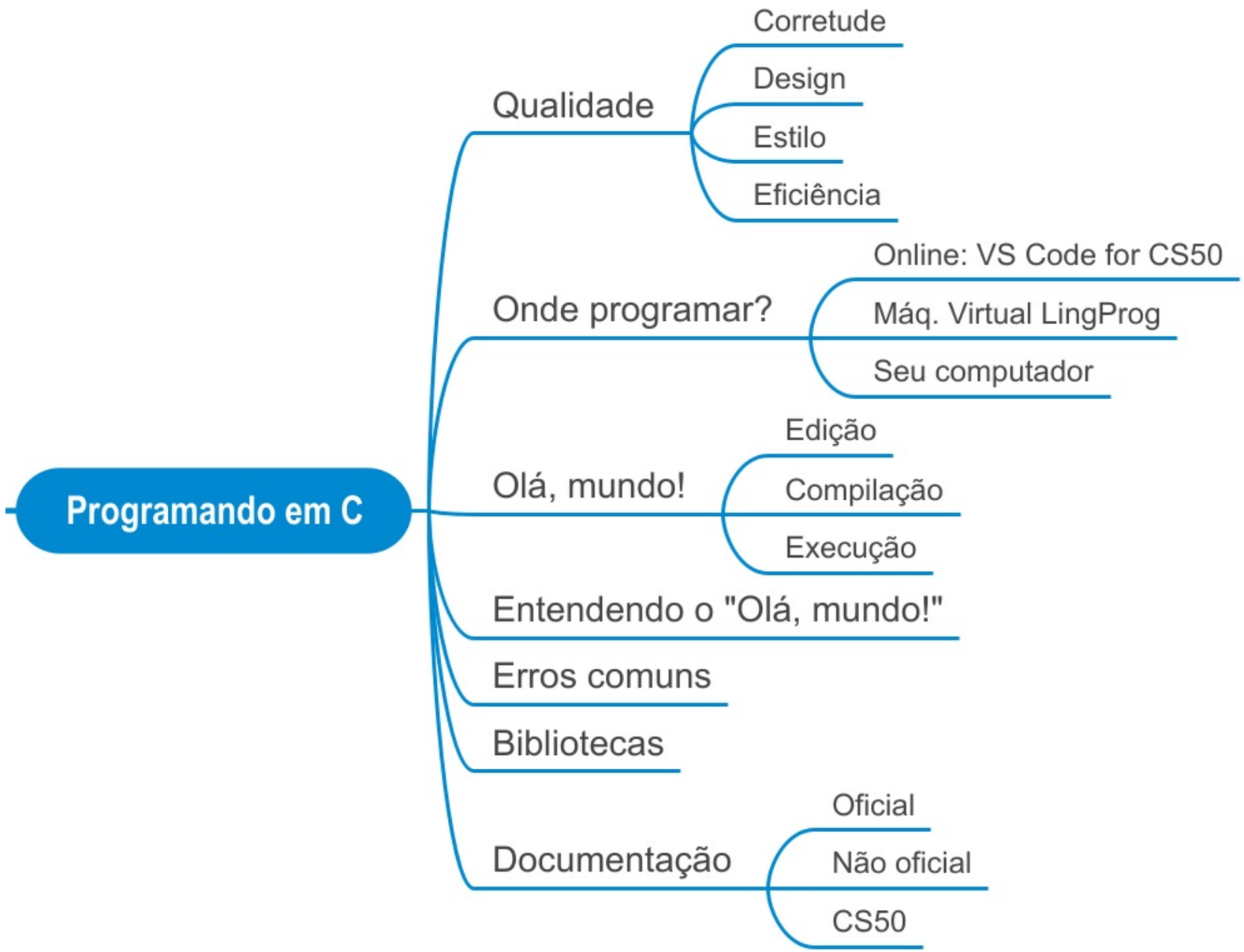
output
(código de máquina)

```
01111111 01000101 01001100 01000110 00000010
00000000 00000000 00000000 00000000 00000000
00000010 00000000 00111110 00000000 00000001
10110000 00000101 01000000 00000000 00000000
01000000 00000000 00000000 00000000 00000000
11010000 00010011 00000000 00000000 00000000
00000000 00000000 00000000 00000000 01000000
00001001 00000000 01000000 00000000 00100100
```

Compiladores para C:

- **GCC (GNU Compiler Collection)**
- **Clang (Clang/LLVM)**

Programando em C



Programando em C: avaliação de seus programas

Você não deve somente escrever códigos, deve escrever bons códigos!

- **Difícil, não se aprende do dia para a noite**
- **Requer muita prática (de programação e de leitura de programas)**

Qualidade do seu código:

Corretude:

Seu programa faz o que deveria fazer?

Design:

Elegância, sofisticação
(subjetivo, aprende-se com o tempo)

Estilo:

Estética, organização, do seu código
(https://cs50.readthedocs.io/style/c/)

Eficiência:

Seu código é rápido o suficiente?

ola.c

```c
#include <stdio.h>

int main(void)
{
    printf("%s\n", "Olá, mundo!");
    return 0;
}
```

```c
#include <stdio.h>
                                int
main(
    void    )
                                {
printf("%s\n",
                                "Olá, mundo!"
                );
    return
0
                                ;
}
```

Programando em C: onde programar?

Editor de texto puro +
Git/GitHub

Recomendado:
Ambiente de Desenvolvimento Integrado
(Integrated Development Environment - IDE)

Caderno, lápis, borracha e apontador!

Ferramentas online:

- **Visual Studio Code for CS50** **(https://cs50.dev)**

Máquina virtual:

- **Computação Raiz: LingProg(https://we.tl/t-4dwWMffWod)**

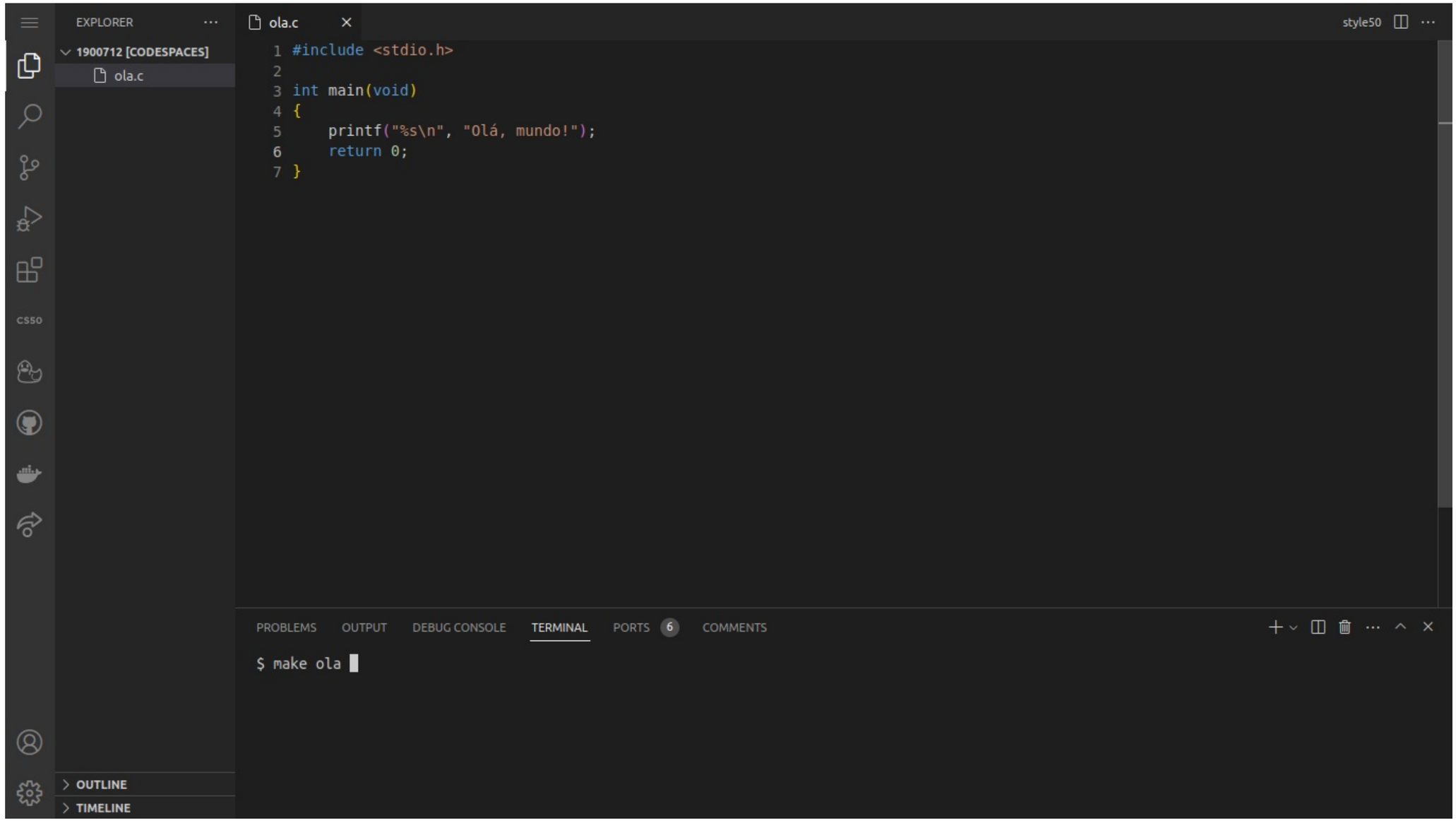
Seu computador Linux/Mac:

- **Visual Studio Code** **(https://code.visualstudio.com)**
- **VSCodium** **(https://vscodium.org)**
- **GCC ou Clang**

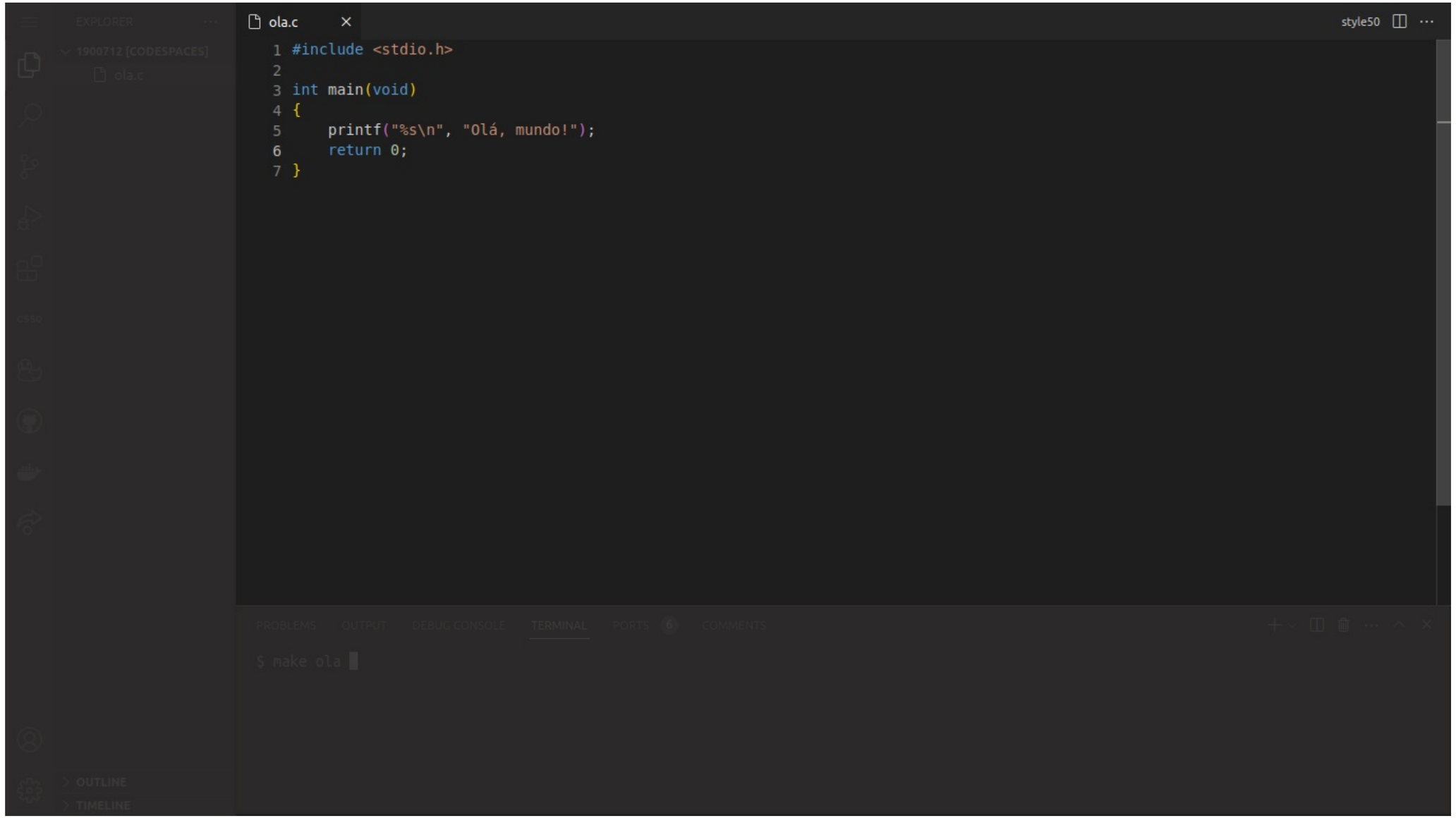
Seu computador Windows:

- **Visual Studio Code** **(https://code.visualstudio.com)**
- **MinGW-w64** **(https://www.mingw-w64.org)**
- **Ubuntu no WSL** **(https://ubuntu.com/wsl)**

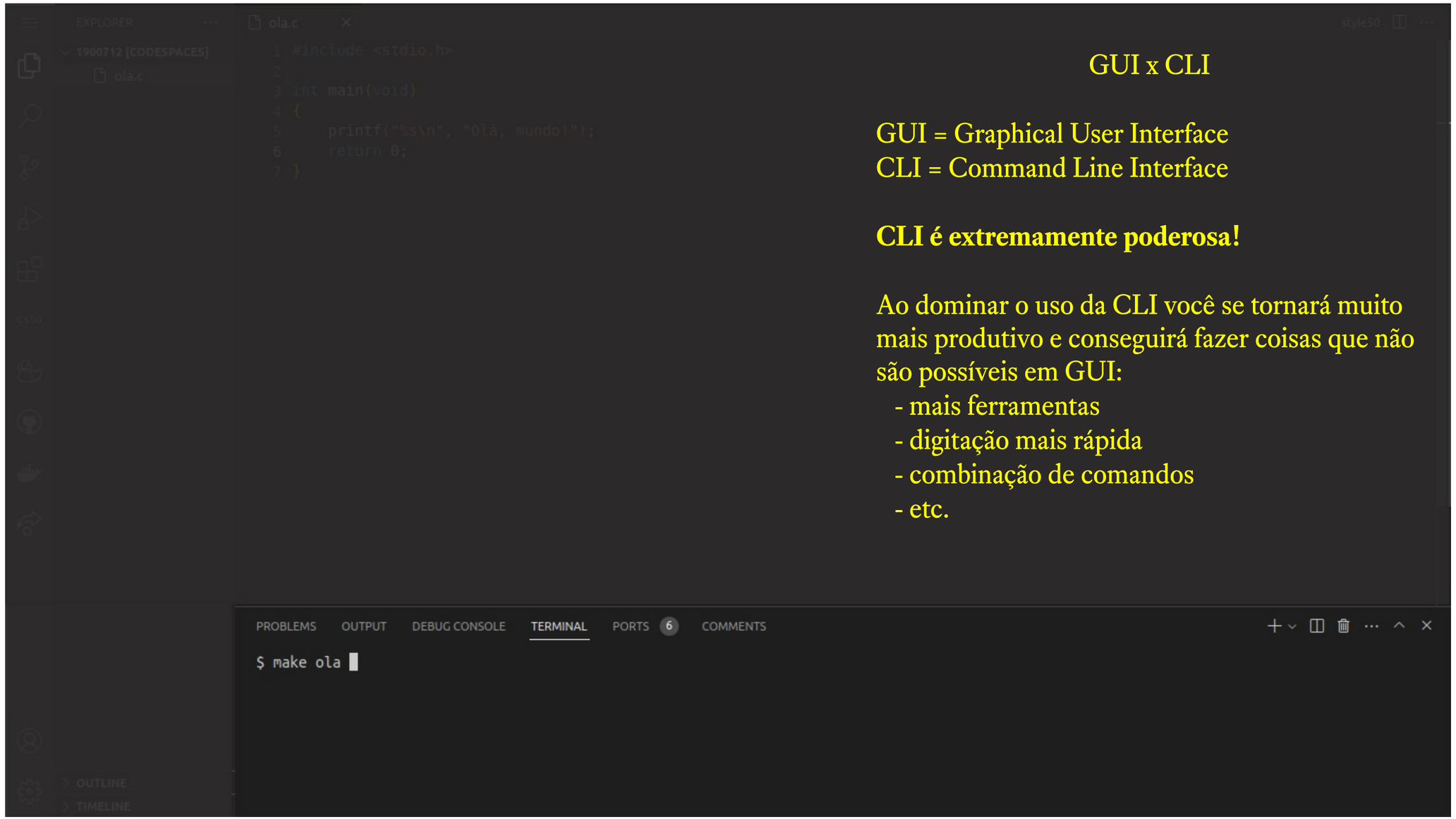
Programando em C: onde programar?



Programando em C: onde programar?



Programando em C: onde programar?



GUI x CLI

GUI = Graphical User Interface
CLI = Command Line Interface

CLI é extremamente poderosa!

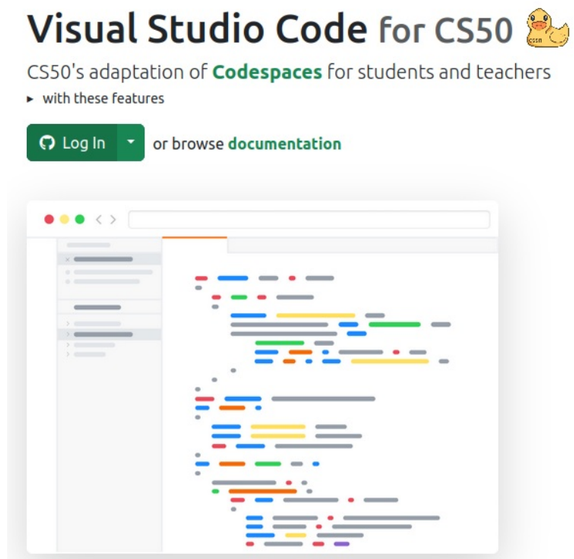
Ao dominar o uso da CLI você se tornará muito mais produtivo e conseguirá fazer coisas que não são possíveis em GUI:

- mais ferramentas
- digitação mais rápida
- combinação de comandos
- etc.

Programando em C: onde programar?

Programando em C: onde programar?

Programando em C: seu primeiro programa



https://cs50.dev

Use o VSCode for CS50 e crie o "Olá, mundo!"

```
$ code ola.c
```

Cria um arquivo de código fonte chamado "ola.c".

Não use caracteres especiais!

Use letras minúsculas sem acentos, e underscore.

Extensão padrão: ".c"

```
$ make ola
```

Compila o código fonte em código de máquina e cria o executável "ola".

Não coloque o ".c" no final!

```
$ ./ola
```

Executa o programa "ola" no diretório atual.

Programando em C: seu primeiro programa

Digite, compile e execute:

```c
#include <stdio.h>

int main(void)
{
    printf("Olá, mundo!\n");
}
```

Note o realce de sintaxe (syntax highlighting) e a indentação.

Se você fizer qualquer alteração no código fonte, precisará compilar novamente!

Programando em C: entendendo o programa

Como seu programa funciona?

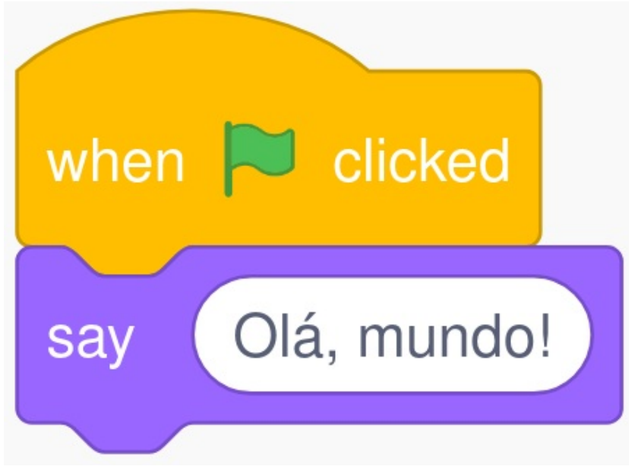
```
#include <stdio.h>

int main(void)
{
    printf("Olá, mundo!\n");
}
```

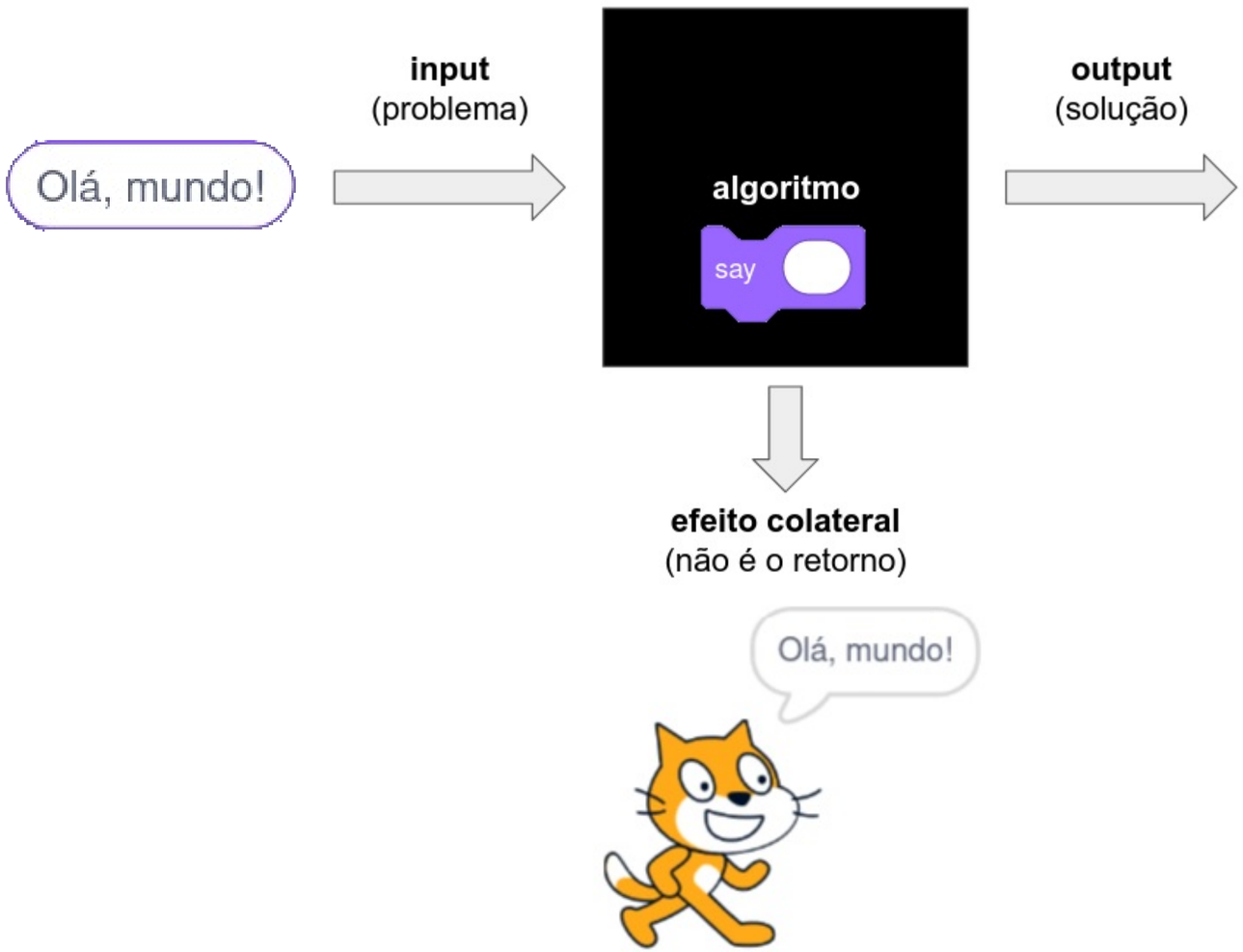
Programamando em C: entendendo o programa

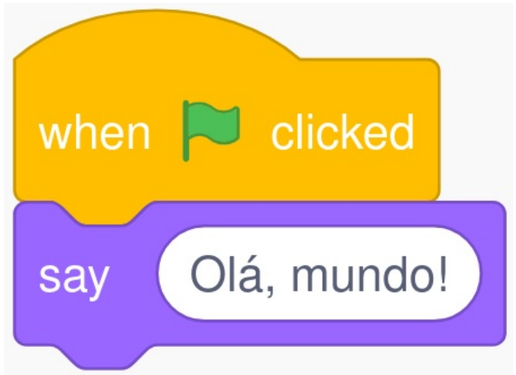
```c
#include <stdio.h>

int main(void)
{
    printf("Olá, mundo!\n");
}
```



Programando em C: entendendo o programa

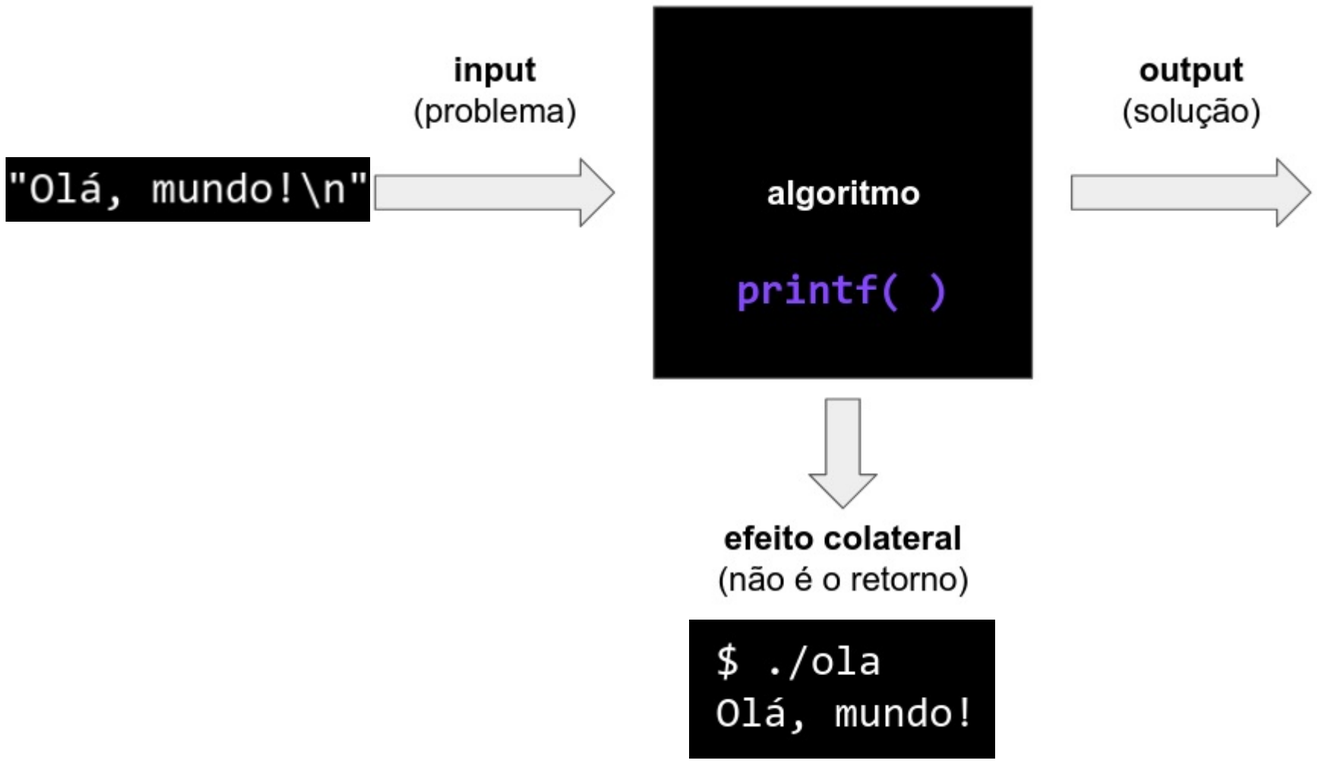


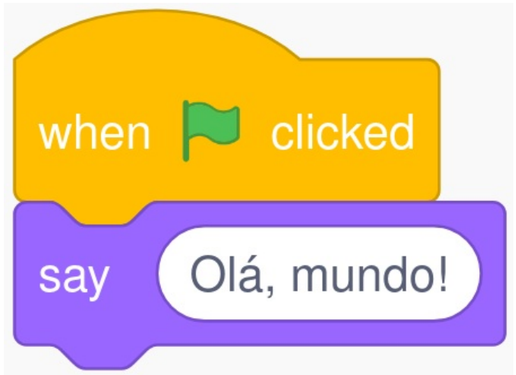



```c
#include <stdio.h>

int main(void)
{
    printf("Olá, mundo!\n");
}
```

Programando em C: entendendo o programa






```c
#include <stdio.h>

int main(void)
{
    printf("Olá, mundo!\n");
}
```

Programando em C: entendendo o programa

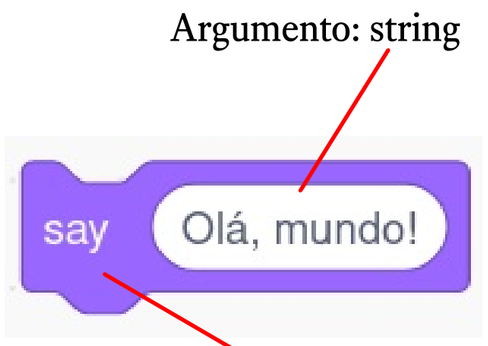
main: **função especial que marca o início de seu programa (veremos posteriormente nessa unidade). Seu programa deve ficar "dentro" da função main, ou seja, dentro das chaves (há exceções, veremos posteriormente).**

```
int main (void)
{

}
```

Programando em C: entendendo o programa

printf: **função para imprimir alguma coisa, de acordo com um formato**
string: **seqüência de 0 ou mais caracteres, entre aspas duplas**
escape sequence: **começa com uma \ e serve para imprimir caracteres não imprimíveis**



Argumento: string

Função

```
printf("Olá, mundo!\n");
```

Argumento: "string"

Finaliza a sentença

\n escape sequence:
imprime uma quebra de linha

Programando em C: erros comuns

- Esquecer de recompilar o programa ao fazer qualquer alteração no código fonte.
- Esquecer o ; no final de uma sentença.
- Colocar ; no final de linhas que não são uma sentença.
- Esquecer o \n quando quiser uma quebra de linha.
- Tentar quebrar a linha dentro de uma string, sem usar \n.
- Escrever studio.h.
- Escrever print.
- Esquecer de fechar as aspas duplas nas strings.
- Usar aspas simples para strings.
- Usar make ola.c.
- Desbalancear delimitadores (chaves, colchetes e parênteses).
- ...

Não se desespere! Leia as mensagens de erro do compilador e veja se alguma coisa te ajuda a corrigir o erro.

Bibliotecas

O que a linha abaixo faz?

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

Por que o sinal de #?
Por que stdio.h?
Por que está entre < >?
Por que não tem ; no final?
O que ocorre se não colocar essa linha?
Por que essa complicação toda?

```c
#include <stdio.h>

int main (void)
{
    printf("Olá, mundo!\n");
}
```

Bibliotecas

Tentativa de compilar o arquivo "ola.c" sem escrever a linha:

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

ola.c

style50

```
1
2
3 int main (void)
4 {
5     printf("Olá, mundo!\n");
6 }
7
```

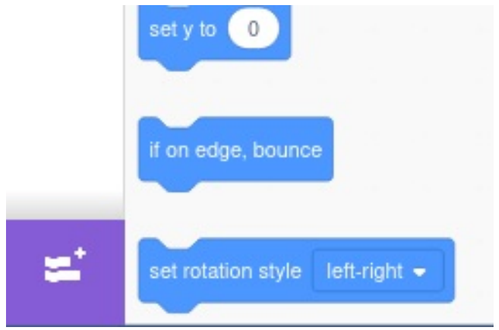
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS 1 COMMENTS

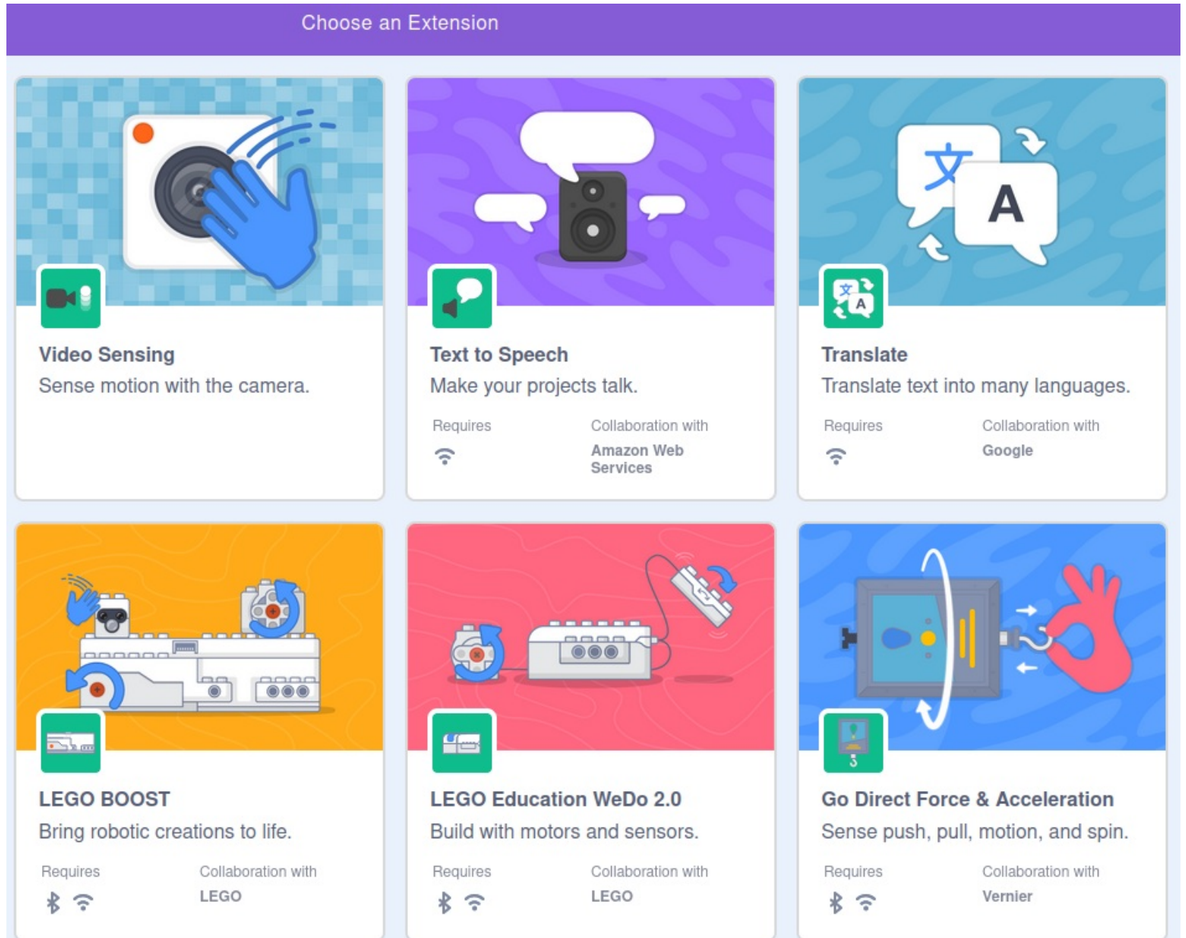
```
$ make ola
ola.c:5:5: error: call to undeclared library function 'printf' with type 'int (const char *, ...)';
ISO C99 and later do not support implicit function declarations [-Wimplicit-function-declaration]
    5 |     printf("Olá, mundo!\n");
      |     ^
ola.c:5:5: note: include the header <stdio.h> or explicitly provide a declaration for 'printf'
1 error generated.
make: *** [<builtin>: ola] Error 1
$
```

O que é esse negócio aí de "printf"?

Bibliotecas

No Scratch, para termos acesso à algumas funções especiais, era necessário clicar no botão "Add Extension" e escolher quais funções especiais iríamos utilizar:





Bibliotecas

Em C é a mesma coisa: muitas funções especiais não fazem parte da linguagem por si mesma, mas estão em bibliotecas que podemos incluir em nossos programas.

Bibliotecas de código: são conjuntos de funções, variáveis e estruturas de dados que outros programadores disponibilizam para usarmos em nossos próprios programas.

Essas bibliotecas (libraries) ficam armazenadas em arquivos chamados de arquivos objeto (.o, .a, .so), em algum lugar do nosso computador. Nós podemos acessar esses arquivos através de uma header file (.h) em nosso programa. Por exemplo:

A library stdio.o é acessada através da header file stdio.h.

Usamos uma header file que "aponta" para a library que tem as funções que estamos querendo usar em nossa programa. A função printf está dentro da library stdio.o. Podemos usar várias headers files ao mesmo tempo.

Tecnicamente a coisa toda é mais complicada... veremos posteriormente.

Bibliotecas

(em algum lugar no seu computador)

stdio.o

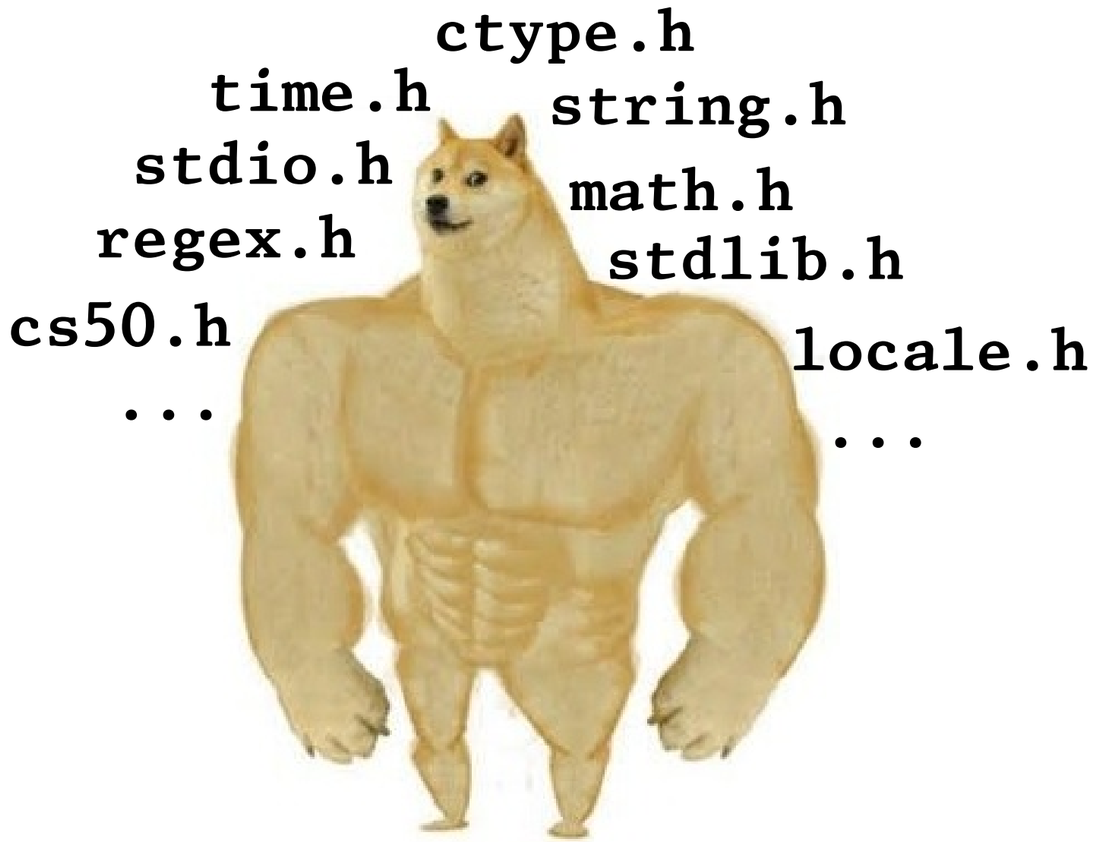
```
#include <stdio.h>

int main (void)
{
    printf("Olá, mundo!\n");
}
```

Tecnicamente a coisa toda é mais complicada... veremos posteriormente.

Bibliotecas: por que usar?

C



C + bibliotecas

Bibliotecas: OK, mas onde estão esses arquivos?

Depende do sistema operacional, da instalação das ferramentas de desenvolvimento, etc. O mais importante é o seguinte: se tudo está instalado e configurado corretamente, você não precisa se preocupar (por enquanto), o compilador faz a coisa certa e inclui a biblioteca no seu programa.

Somente como curiosidade:

libc é a biblioteca padrão (standard library) da linguagem C. Contém várias outras bibliotecas em seu interior.

```
[abrantesasf@cosmos ~]$ ls -lh /usr/include/stdio.*
-rw-r--r-- 1 root root 31K jul  6  2022 /usr/include/stdio.h

[abrantesasf@cosmos ~]$ ls -lh /usr/lib/x86_64-linux-gnu/libc.*
-rw-r--r-- 1 root root 5,8M jul  6  2022 /usr/lib/x86_64-linux-gnu/libc.a
-rw-r--r-- 1 root root  283 jul  6  2022 /usr/lib/x86_64-linux-gnu/libc.so
-rw-r--r-- 1 root root 2,2M jul  6  2022 /usr/lib/x86_64-linux-gnu/libc.so.6


[abrantesasf@cosmos ~]$ cp /usr/lib/x86_64-linux-gnu/libc.a .

[abrantesasf@cosmos ~]$ ar x libc.a stdio.o

[abrantesasf@cosmos ~]$ ls -lh stdio.o
-rw-r--r-- 1 abrantesasf abrantesasf 1,3K ago 12 23:16 stdio.o
```

Bibliotecas: agora é possível responder

O que a linha abaixo faz?

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

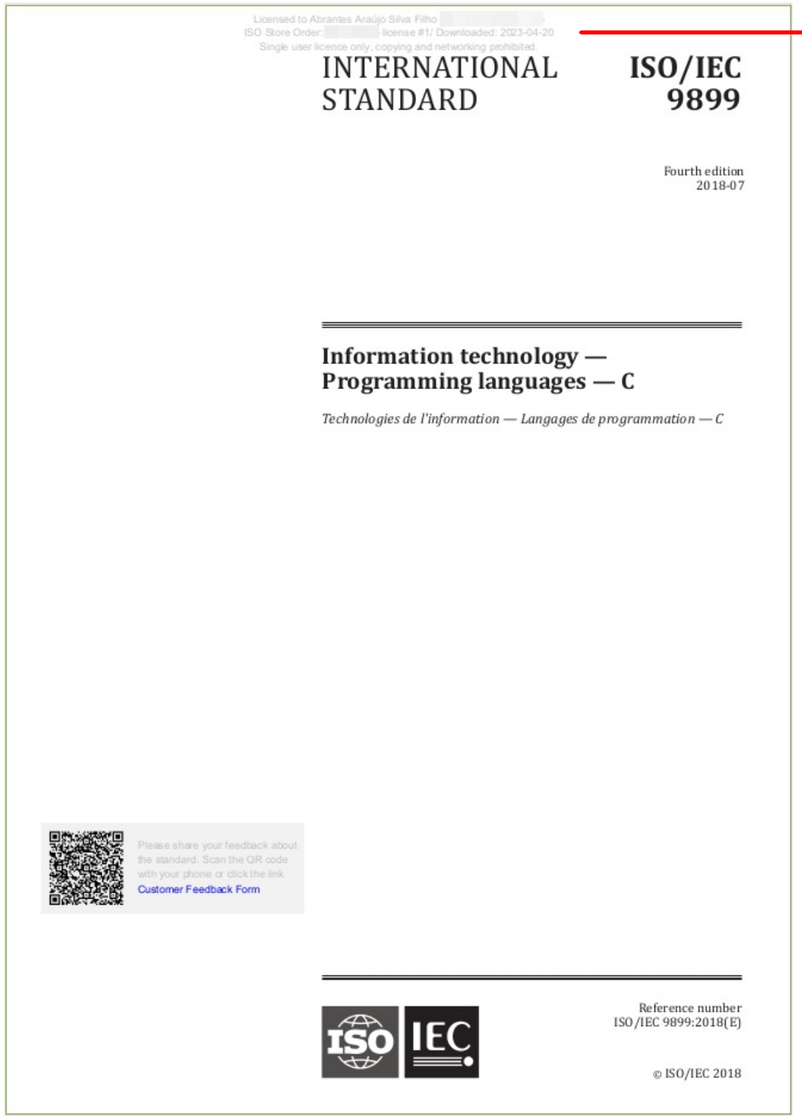
Por que o sinal de #?
Por que stdio.h?
Por que está entre < >?
Por que não tem ; no final?
O que ocorre se não colocar essa linha?
Por que essa complicação toda?

```c
#include <stdio.h>

int main (void)
{
    printf("Olá, mundo!\n");
}
```

Documentação: como saber quais bibliotecas existem?

A linguagem C é padronizada e tem um documento oficial:



INTERNATIONAL STANDARD

ISO/IEC 9899

Fourth edition
2018-07

Information technology — Programming languages — C

Technologies de l'information — Langages de programmation — C

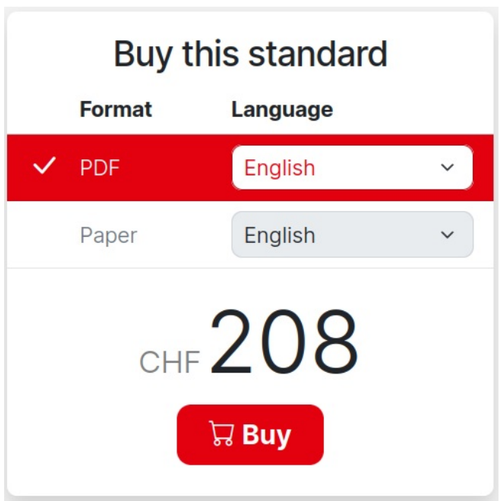
Please share your feedback about the standard. Scan the QR code with your phone or click the link
Customer Feedback Form

Reference number
ISO/IEC 9899:2018(E)

Licensed to Abrantes Araújo Silva Filho
ISO Store Order: license #1/ Downloaded: 2023-04-20
Single user licence only, copying and networking prohibited.

O padrão oficial atual da linguagem C, chamado de C17, é vendido no site da ISO. Em 2023, o mais recente é o padrão ISO/IEC 9899:2018, disponível em:

www.iso.org/standard/74528.html



Documentação: como saber quais bibliotecas existem?

É possível acessar gratuitamente os PDFs dos rascunhos dos padrões na página do grupo de trabalho da ISO para a linguagem C: www.open-std.org/JTC1/SC22/WG14

ISO/IEC JTC1/SC22/WG14 - C — Mozilla Firefox

ISO/IEC JTC1/SC22/ ×

https://www.open-std.org/JTC1/SC22/WG14/ 100%

Welcome to the official home of

JTC1/SC22/WG14 - C

2021-11-25: projects | documents | contributing | internals | meetings | contacts

ISO/IEC JTC1/SC22/WG14 is the international standardization working group for the programming language C.

The current C programming language standard (C17) ISO/IEC 9899 was adopted by ISO and IEC in 2018.

To obtain the international standard, please contact your national member body.

Work on projects and their milestones include:

- 9899: Programming Language C
- Defect Reports and Record of Response
- TR 18037: Embedded C
- TR 19769: Extensions for the programming language C to support new character data types
- TR 24731-1: Extensions to the C Library Part I: Bounds-checking interfaces
- TR 24731-2: Extensions to the C Library Part II: Dynamic allocation functions
- TR 24732: Extensions for the programming language C to support decimal floating point arithmetic
- WG14 has finished revising the C standard, under the name C11. A charter for the revision of the standard describes the rules for what has been done.
- TR 24747: Programming language C - Extensions for the C language library to support mathematical special functions
- The rationale for the C99 standard is available.

Other information available is:

- The WG document register including the documents
- New WG wiki (protected, only for members)
- WG internal information (protected, only for members)
- Information on WG meetings
- WG14 Business Plan and Convener's Report 2012
- Working Group Standing Document 1, mailings and meetings information
- Working Group Standing Document 2, Informal Study Group Organization Information
- Working Group Standing Document 3, Partial list of proposals that did not fit into the former DR process for C11
- ISO/IEC 2382:2015 Information technology — Vocabulary (freely available standard)
- Contacts

If you want further information, or want to participate, please contact your national member body or one of the contact addresses of the WG.

2021-11-25: projects | documents | contributing | internals | meetings | contacts

This page is sponsored by DTU. HTML design by Keld Simonsen. Comments welcome!

C - Project status and milestones

2023-04-05: home | projects | documents | contributing | internals | meetings | contacts

ISO/IEC 9899 - Revision of the C standard

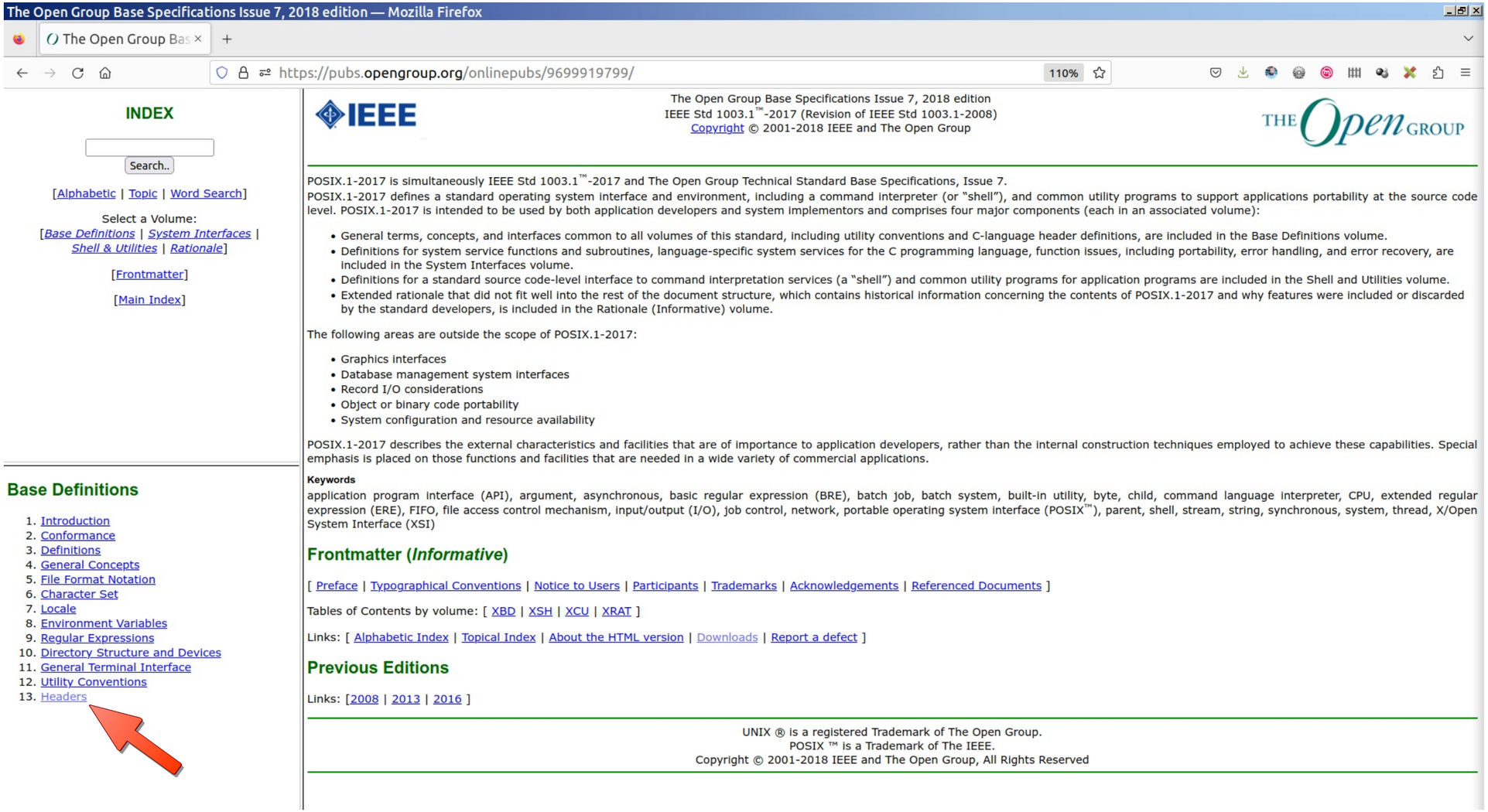
The primary output of WG14 is ISO/IEC 9899, the C Standard. The following is a list of revisions to ISO/IEC 9899 that the committee has produced:

Revision	ISO publication	Similar draft
C2x	Not available	N3096 [2023-04-02]
C17	ISO/IEC 9899:2018	N2310 [2018-11-11] (early C2x draft)
C11	ISO/IEC 9899:2011	N1570 [2011-04-04]
C99	ISO/IEC 9899:1999	N1256 [2007-09-07]
C89	ISO/IEC 9899:1990	Not available

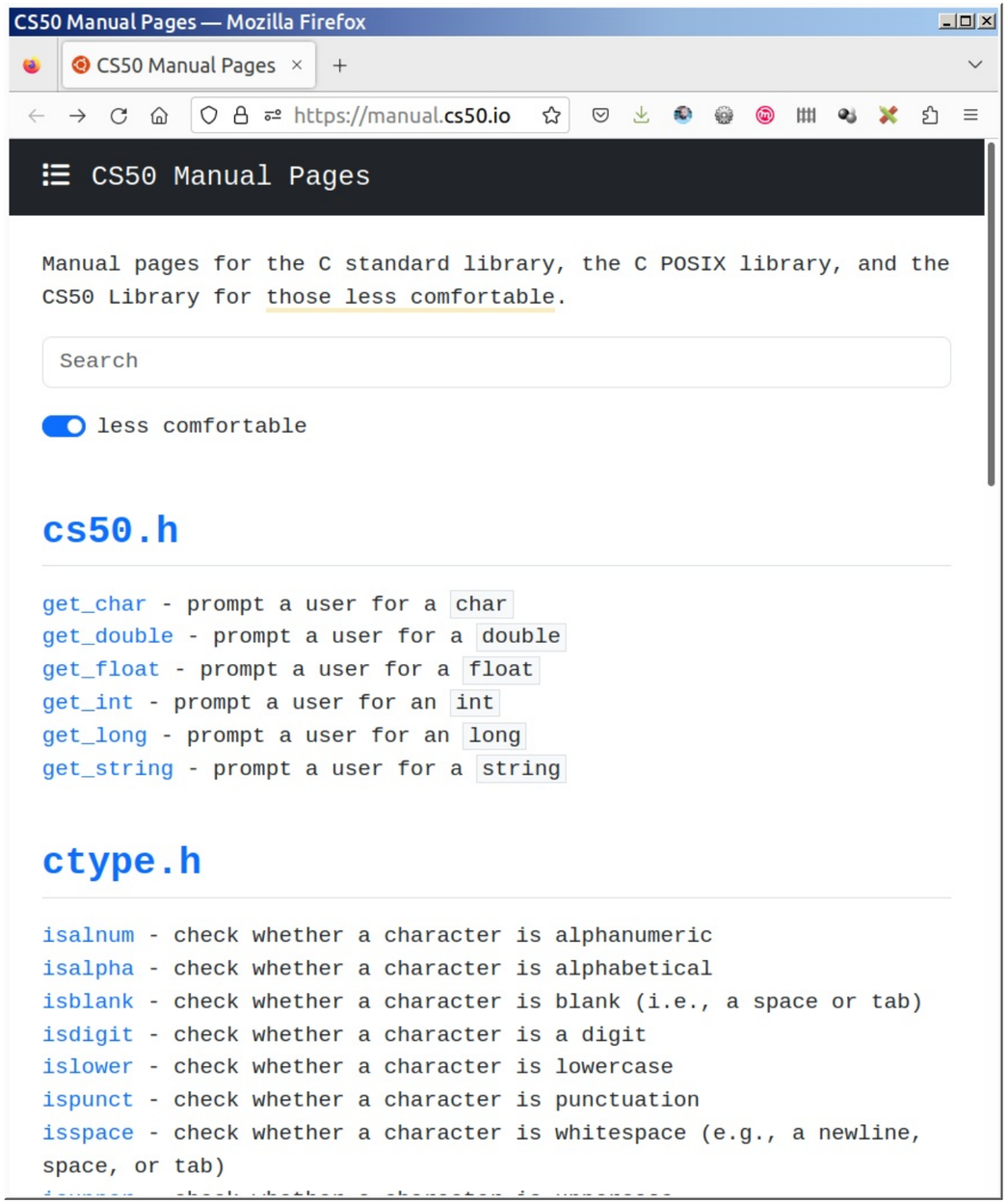
Documentação: como saber quais bibliotecas existem?

No "The Open Group" é possível acessar a documentação das header files:

https://pubs.opengroup.org/onlinepubs/9699919799/



Documentação: como saber quais bibliotecas existem?



A Universidade de Harvard pegou as header files disponíveis no site do The Open Group e simplificou a formatação e o texto para alunos iniciantes na CS50 (a versão completa também está disponível). É essa documentação que você deve consultar periodicamente!

https://manual.cs50.io

Obrigado CS50!

Documentação: como saber quais bibliotecas existem?

Na documentação preparada pela CS50 você tem 2 opções:

less confortable

stdio.h

fclose - close a file
fopen - open a file
fprintf - print to a file
fread - read bytes from a file
fscanf - get input from a file
fwrite - write bytes to a file
printf - print to the screen
scanf - get input from a user
sprintf - print to a string

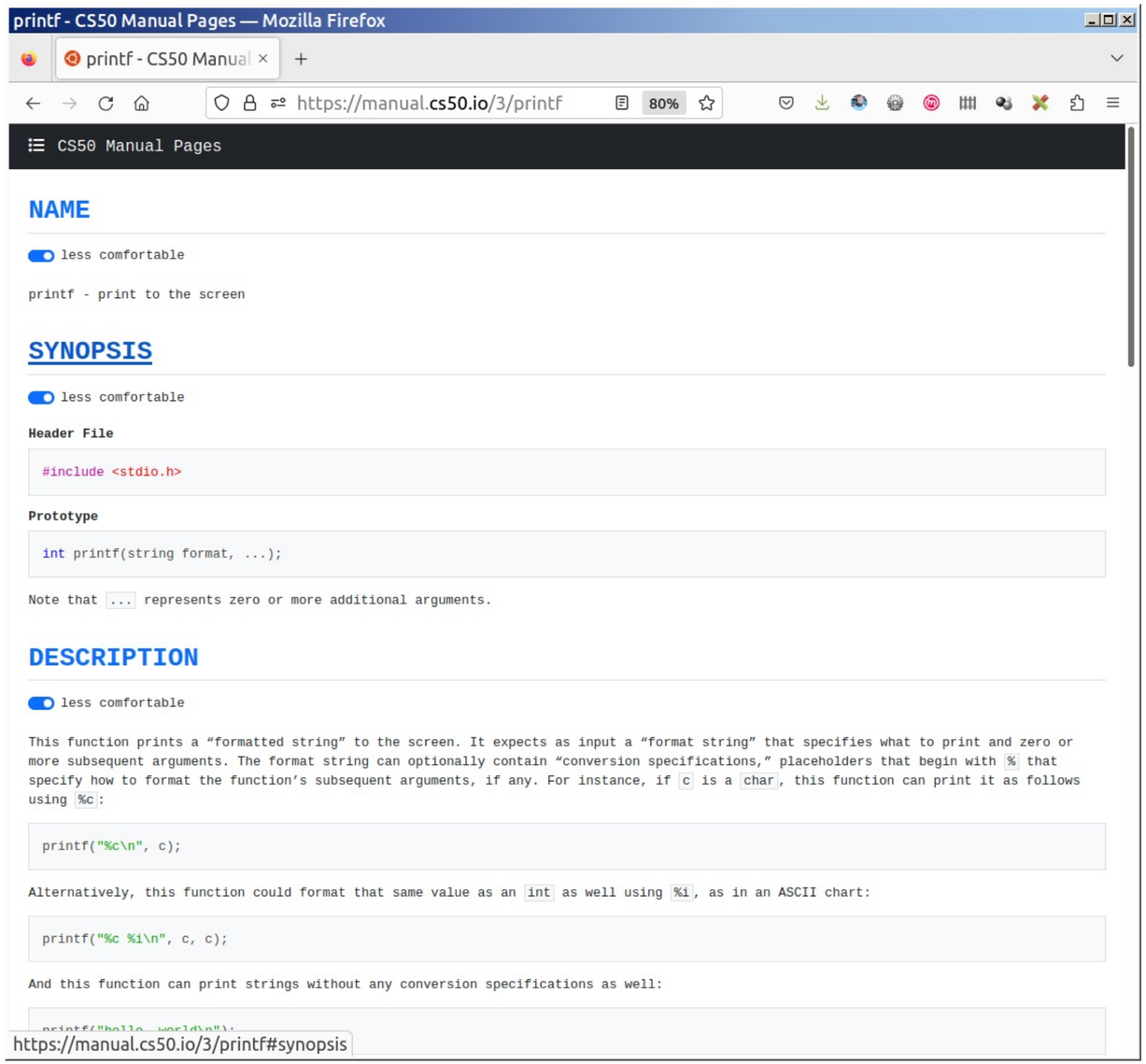
more confortable

stdio.h

__fbufsize - interfaces to stdio FILE structure
__flbf - interfaces to stdio FILE structure
__fpending - interfaces to stdio FILE structure
__fpurge - purge a stream
__freadable - interfaces to stdio FILE structure
__freading - interfaces to stdio FILE structure
__fsetlocking - interfaces to stdio FILE structure
__fwritable - interfaces to stdio FILE structure
__fwriting - interfaces to stdio FILE structure
_flushlbf - interfaces to stdio FILE structure
addmntent - get filesystem descriptor file entry
asprintf - print to allocated string
clearerr - check and reset stream status
clearerr_unlocked - nonlocking stdio functions
ctermid - get controlling terminal name
dprintf - formatted output conversion
endmntent - get filesystem descriptor file entry
fclose - close a stream
fcloseall - close all open streams
fdopen - stream open functions
feof - check and reset stream status
feof_unlocked - nonlocking stdio functions
ferror - check and reset stream status
ferror_unlocked - nonlocking stdio functions
fflush - flush a stream
fflush_unlocked - nonlocking stdio functions
fgetc - input of characters and strings
fgetc_unlocked - nonlocking stdio functions
fgetgrent - get group file entry

Documentação: como saber quais bibliotecas existem?

A documentação de cada função também tem opções com mais e menos detalhes:



Name
Library
Synopsis
Description
Return values
Examples
Attributes
Standards
History
Caveats/Notes
Bugs
See also
Colophon

Documentação: como saber quais bibliotecas existem?

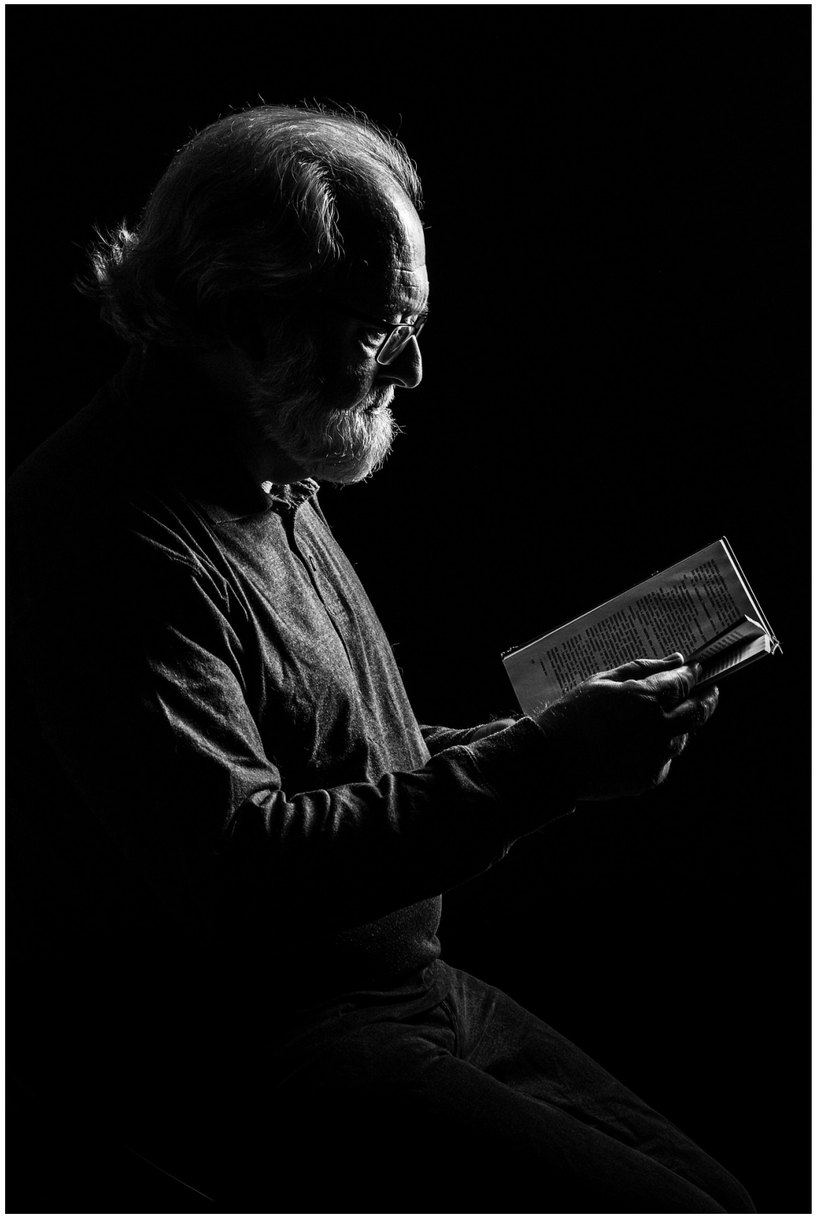


Foto: fotoerich, no Pixabay
(https://pixabay.com/photos/man-reading-education-portrait-6530416/)

Em resumo:

RTFM

(Read The Fucking Manual)

List of similar initialisms [edit]

- RTBM – "read the bloody manual"[6]
- RTFA – "read the fucking/featured article"– common on news forums such as Fark[7] and Slashdot, where using "TFA" instead of "the article" has become a meme[*citation needed*]
- RTDA – "read the damn article"
- RTDM – "read the damn menu"
- RTDM – "read the damn manual"
- WABM – "write a better manual" – an answer complaining that the manual is not written well[8]
- RTFD – "read the fucking documentation"
- RTFE – "read the fucking error"
- RTFM41 – "read the fucking manual for once"
- RTFW – "read the fucking wiki"
- RTFC – "read the fucking chart" – a response that physicians give to coders who submit inappropriate queries
- RTFS – "read the fucking source" or "read the fucking standard" or "read the fucking syllabus"[9]
- RTFB – "read the fucking binary"[10]
- RTFI – "read the fussy instructions"
- RTFQ – "read the fucking question"
- RTMS – "read the manual, stupid"
- STFW – "search the fucking web"
- GIYF – "Google is your friend"
- JFGI – "just fucking Google it"
- DYOR – "do your own research"
- LMGTFY, LMGT4U – "let me Google that for you"

RTFM na Wikipedia: https://en.wikipedia.org/wiki/RTFM

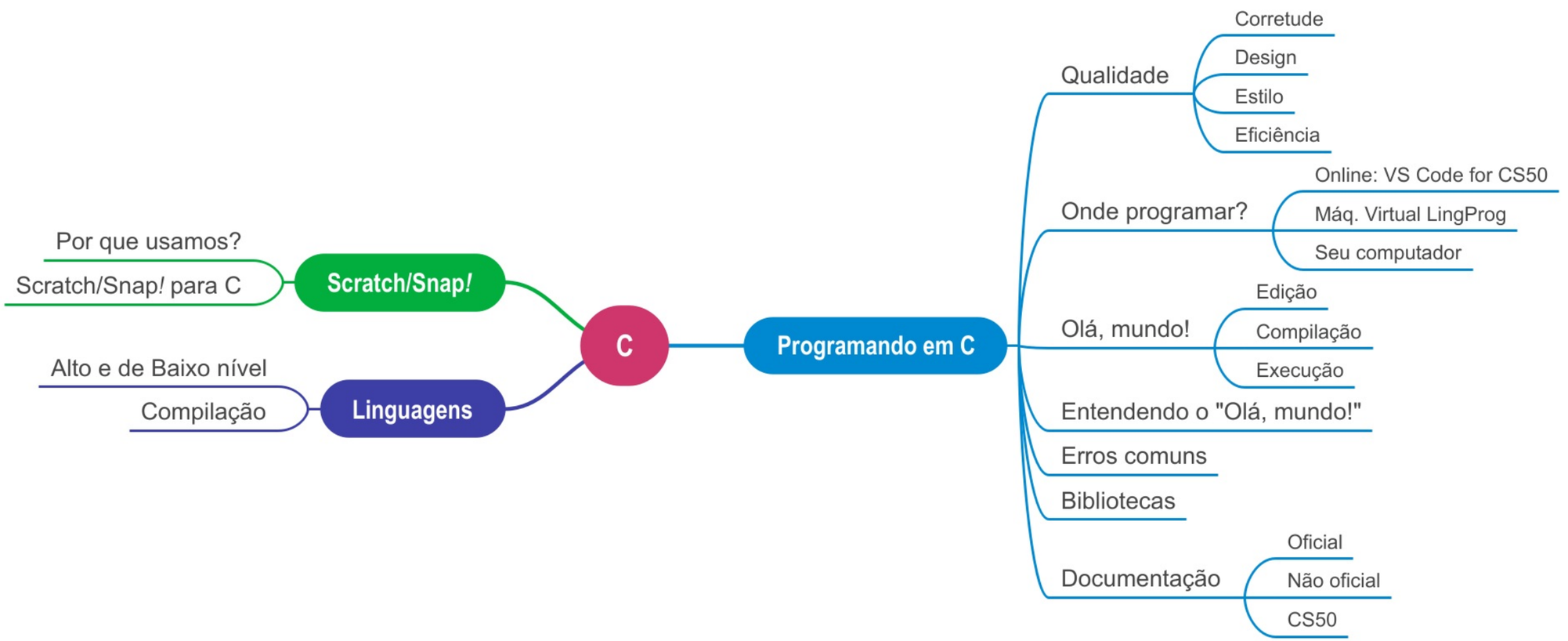
Documentação: outras

Existem muitas outras fontes de documentação para a linguagem C, tais como:

- **Livros**
- **Tutoriais na Internet**
- **Canais no YouTube**
- **Cursos online**
- **...**

Vou citar algumas das melhores quando for apropriado.

Resumo


C
Scratch/Snap!
Por que usamos?
Scratch/Snap! para C
Linguagens
Alto e de Baixo nível
Compilação
Programando em C
Qualidade
Corretude
Design
Estilo
Eficiência
Onde programar?
Online: VS Code for CS50
Máq. Virtual LingProg
Seu computador
Olá, mundo!
Edição
Compilação
Execução
Entendendo o "Olá, mundo!"
Erros comuns
Bibliotecas
Documentação
Oficial
Não oficial
CS50