

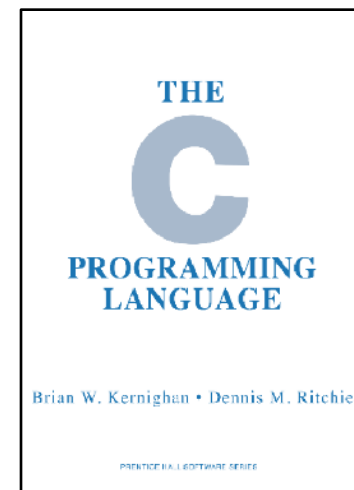
Laboratório de Programação I



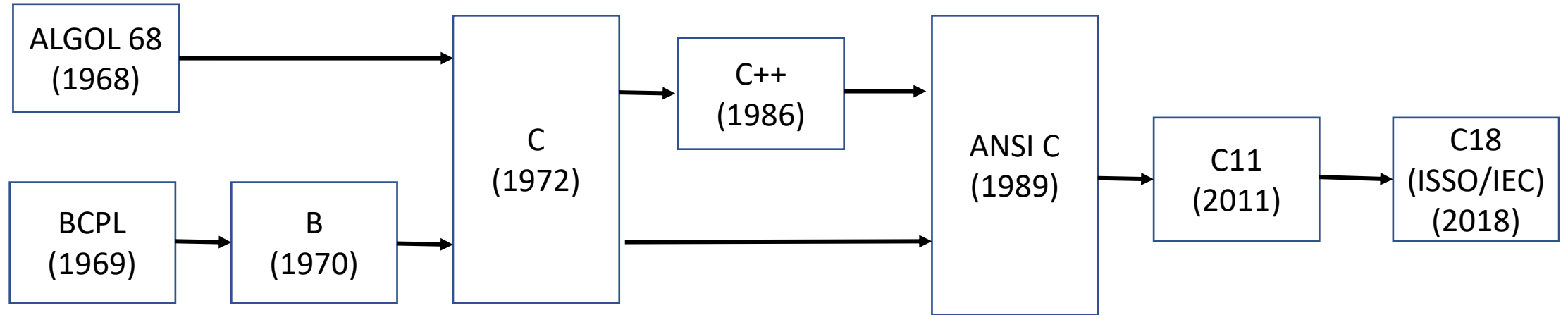
Prof.: Claudio Junior (claudiojns@ufba.br)
Introdução à linguagem C

Histórico da Linguagem C

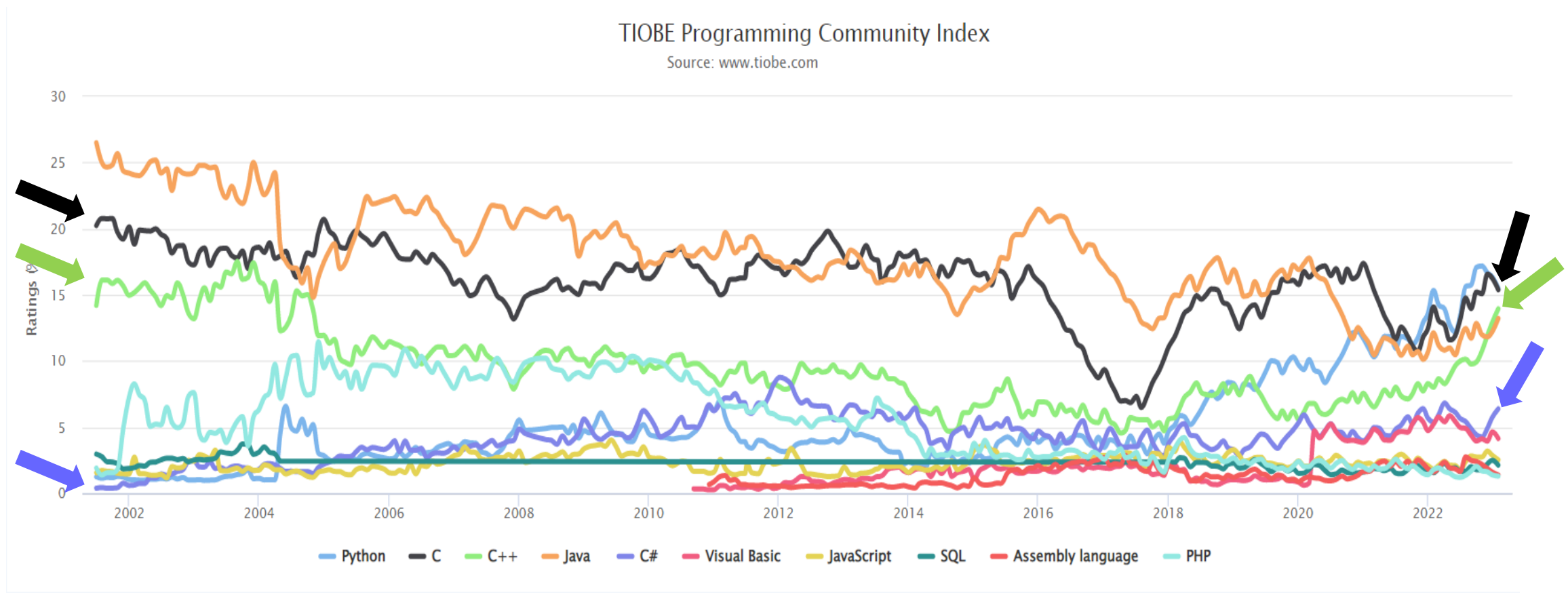
- Dennis Ritchie em 1972 (DEC PDP-11, Bell Laboratories);
- Evolução da Linguagem B, adaptação da Linguagem BCPL (Ken Thompson), e Algol 68;
- Foco no desenvolvimento de Sistemas Operacionais (SO) e compiladores;
- Associada ao SO UNIX;
- The C Programming Language - Kernighan & Ritchie, 1978;
- Linguagem de propósito geral (1980);
- American National Standards Institute (ANSI) – Padrão ANSI-C em 1983 (1989):
 - Incorporação de uma biblioteca padrão, presente em todas as versões da linguagem, que fornece funções de acesso ao SO, entrada e saída formatada, alocação de memória, manipulação de strings (cadeia de caracteres), etc.



Histórico da Linguagem C













Linguagens de Programação



Fonte: <https://www.tiobe.com/tiobe-index/>

Acesso em: 14 de fevereiro de 2022

Linguagens de Programação

Feb 2023	Feb 2022	Change	Programming Language	Ratings
1	1		 Python	15.49%
2	2		 C	15.39%
3	4	↑	 C++	13.94%
4	3	↓	 Java	13.21%
5	5		 C#	6.38%
6	6		 Visual Basic	4.14%
7	7		 JavaScript	2.52%
8	10	↑	 SQL	2.12%
9	9		 Assembly language	1.38%
10	8	↓	 PHP	1.29%
11	11		 Go	1.11%
12	13	↑	 R	1.08%
13	14	↑	 MATLAB	0.99%
14	15	↑	 Delphi/Object Pascal	0.95%
15	12	↓	 Swift	0.93%
16	16		 Ruby	0.83%
17	19	↑	 Perl	0.79%
18	22	↑↑	 Scratch	0.76%
19	17	↓	 Classic Visual Basic	0.74%
20	24	↑↑	 Rust	0.70%

35,71%

Fonte: <https://www.tiobe.com/tiobe-index/>
Acesso em: 14 de fevereiro de 2022

Linguagens de Programação

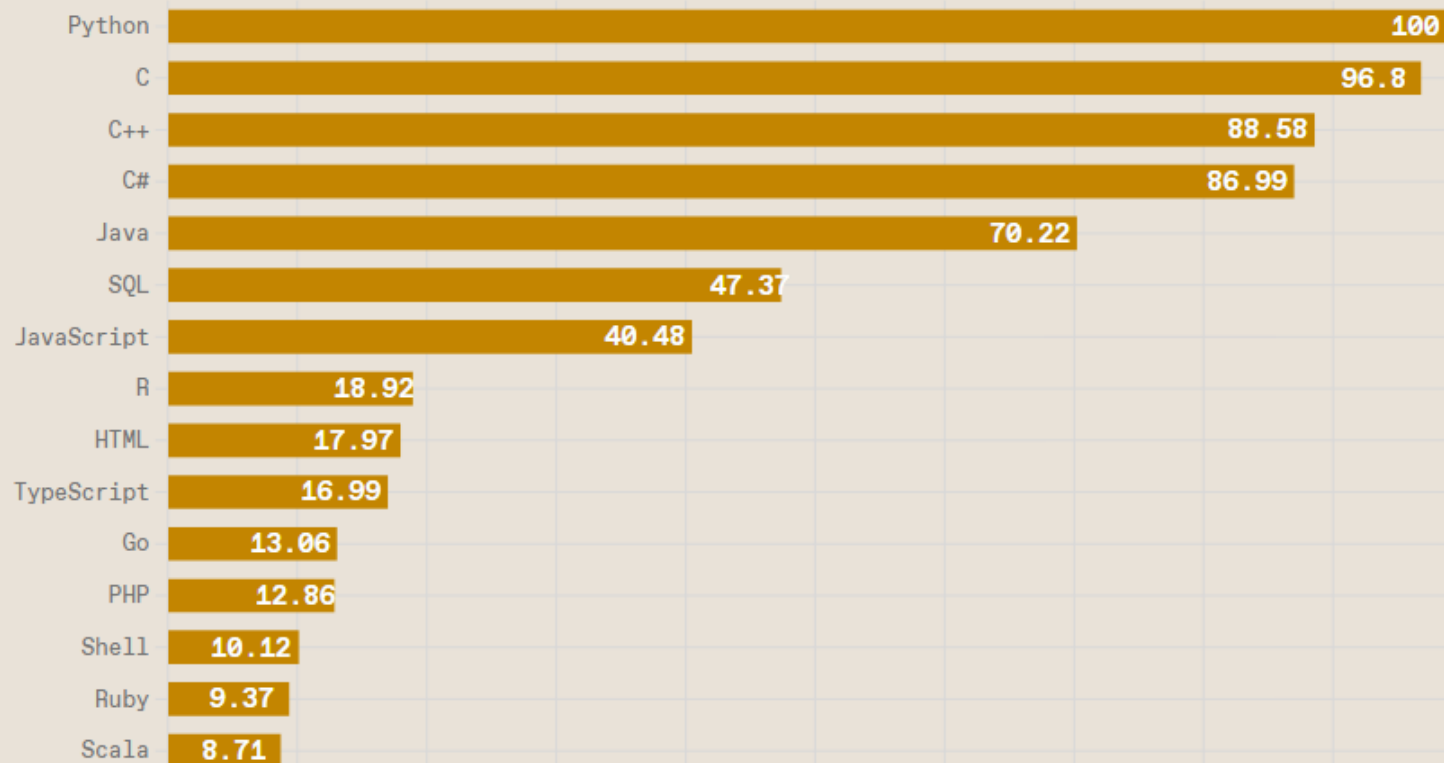
Top Programming Languages 2022

Click a button to see a differently weighted ranking

Spectrum

Jobs

Trending



Language Rank	Types	Spectrum Ranking
1. C		100.0
2. Java		98.1
3. Python		98.0
4. C++		95.9
5. R		87.9
6. C#		86.7
7. PHP		82.8
8. JavaScript		82.2
9. Ruby		74.5
10. Go		71.9

Ranking 2016

Fonte: <https://spectrum.ieee.org/top-programming-languages-2022>

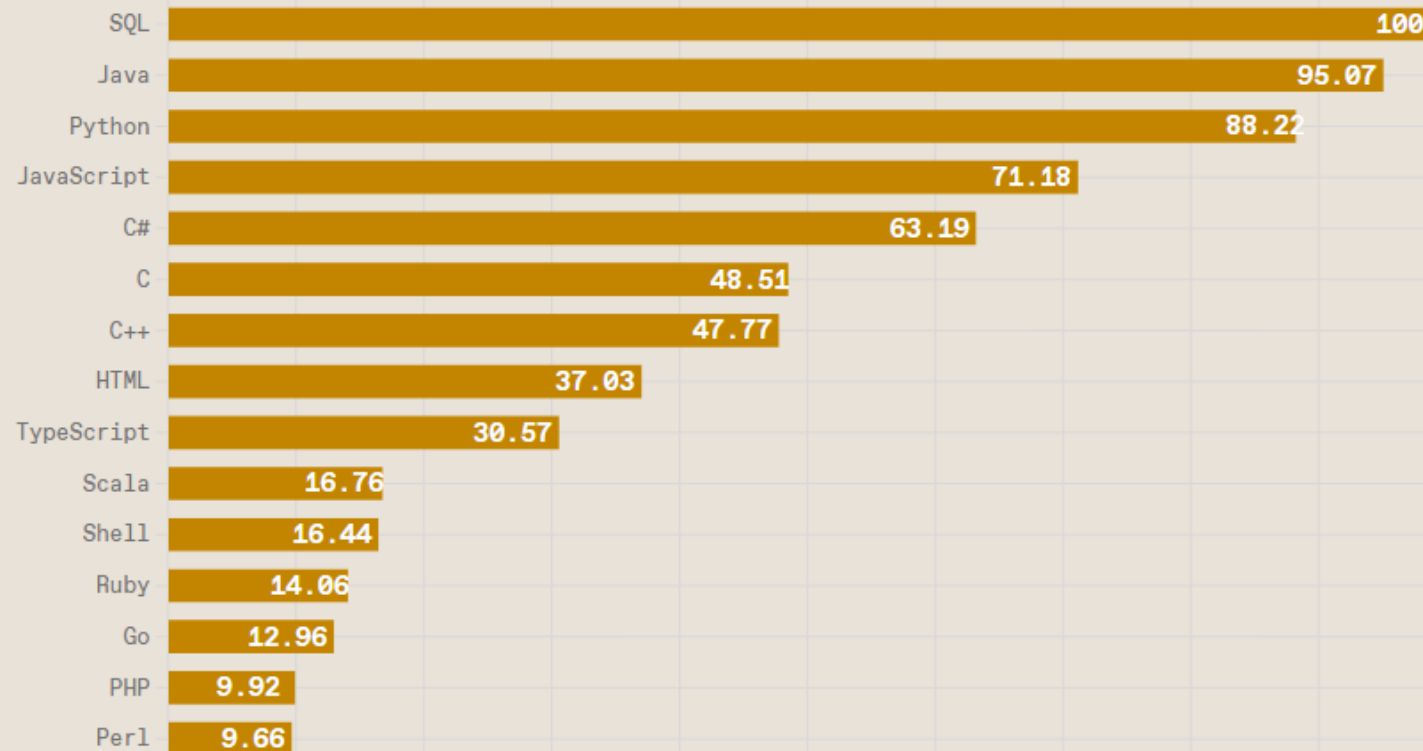
Acesso em: 14 de fevereiro de 2022

Linguagens de Programação

Top Programming Languages 2022

Click a button to see a differently weighted ranking

Spectrum **Jobs** Trending



Fonte: <https://spectrum.ieee.org/top-programming-languages-2022>

Acesso em: 14 de fevereiro de 2022

Linguagens de Programação

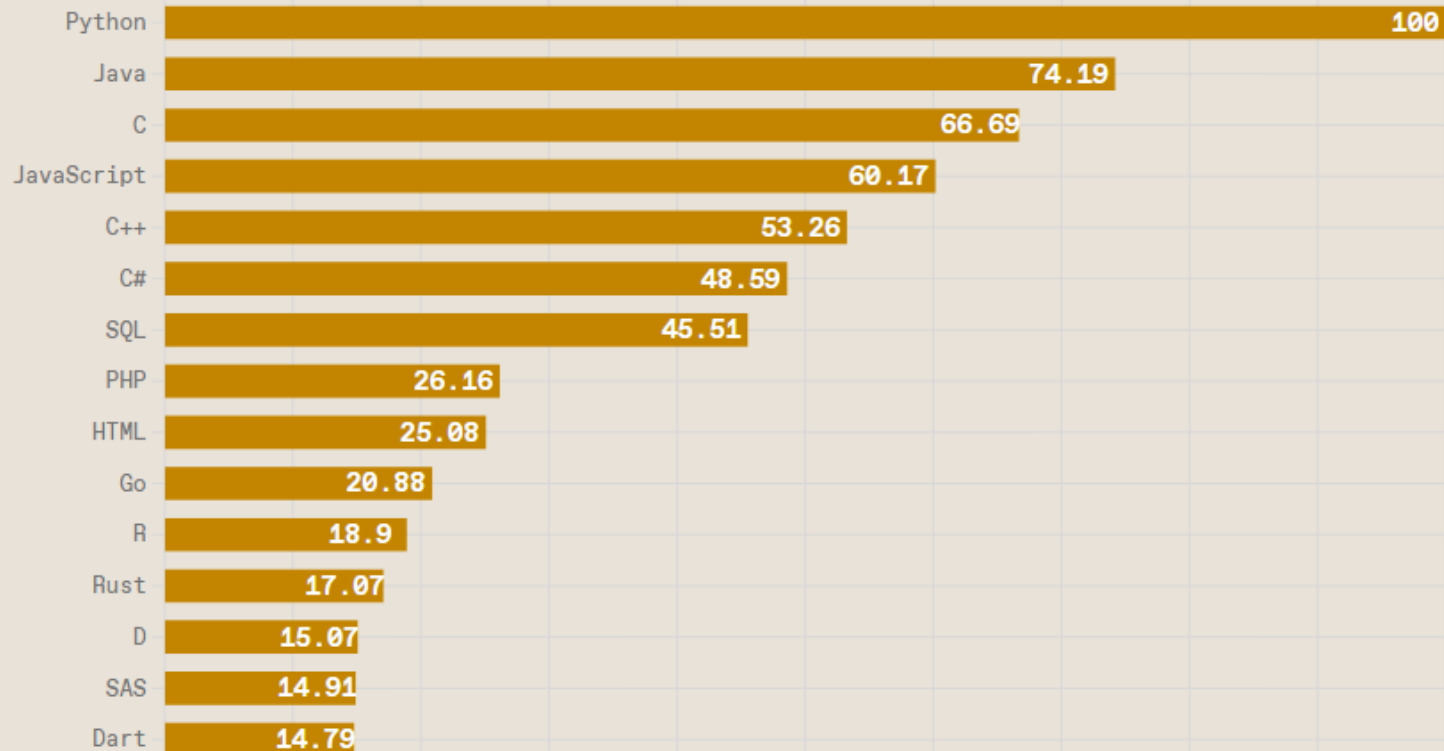
Top Programming Languages 2022

Click a button to see a differently weighted ranking

Spectrum

Jobs

Trending



Fonte: <https://spectrum.ieee.org/top-programming-languages-2022>

Acesso em: 14 de fevereiro de 2022

Características da Linguagem C

C,C++
(Médio nível)

Linguagem estruturada
(Funções, Blocos de
código);

Compilada

Case sensitive
(else # Else # ELSE)

Próxima ao hardware

Manipulação de bits,
bytes

Portável (Mac, PC,
Linux, Windows,
embarcados, etc)

Não é rica em Tipo de
Dados (inteiro,
caractere, real)

Não efetua verificações
de tipo de dados
(fracamente tipada)

Poucas palavras
reservadas

Uso de {} para agrupar
comandos de uma
estrutura lógica (if-else,
do-while, for, etc.)

Para programadores
(poucas restrições,
poucas palavras-chave,
funções isoladas);

SO, Embarcados,
Compiladores, Tempo
real

Palavras reservadas C ANSI

asm	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 e C++

C	C++
Tradicional	Extensão do C
Procedural	Orientada a Objetos e a procedimentos
Baixo/médio nível	Pode atuar em alto nível
Criação de Sistemas Operacionais	Aplicações diversas
.c	.cpp
	++ : incremento, acréscimo, algo a mais, “C com classes” (Bjarne Strous)

Estrutura básica de um programa C

Diretivas

Declaração de variáveis/constantes globais;

Prototipação das funções;

*tipo_dado main (lista_parâmetros) /*função principal e obrigatória*/*

{

Declaração de variáveis/constantes locais;

Comandos;

Estruturas de controle (Seleção e/ou repetição);

Comentários;

Chamada a outras funções;

}

tipo_dado func1 (lista_parâmetros)

{

Bloco de comandos;

}

...

tipo_dado funcn (lista_parâmetros)

{

Bloco de comandos;

}

Diretivas

- Diretivas de compilação;
- Localizadas quase sempre no **início** dos arquivos-fonte, com o prefixo **#** seguido pelo nome da diretiva e seus argumentos. **Não** usa **ponto e vírgula** ao final da declaração;
- Diretiva **INCLUDE**:
 - Incorpora ao arquivo-fonte o conteúdo de outro arquivo passado como argumento;
 - Permite o uso de arquivos criados pelo **usuário**, visando **reutilização**, ou fornecidos com o **compilador** (suprir **definições** e **declarações frequentes**)
 - `#include <stdio.h>`
 - `#include <iostream>`

Exemplos #include – Padrão ANSI C

ARQUIVO	FINALIDADE
assert.h	Contém a macro "assert" que é usada para incluir diagnóstico em programas
ctype.h	Declara funções para testar caracteres
errno.h	Define o número de erro do sistema
float.h	Define os valores em ponto flutuante específicos do compilador
limits.h	Define tipos de valores-limites específicos do compilador
locale.h	Utilizada para adaptar as diferentes convenções culturais
math.h	Define as funções matemáticas e macros usadas para operações matemáticas em linguagem C
setjmp.h	Provê uma forma de evitar a seqüência normal de chamada e retorno de função profundamente aninhada
signal.h	Provê facilidades para manipular condições excepcionais que surgem durante a execução, como um sinal de interrupção de uma fonte externa ou um uso na execução
stdarg.h	Define macros para acessar argumentos quando uma função usa um número variável de argumentos
stddef.h	Define funções, constantes, macros e estruturas usadas para operações padrões de entrada e saída
stdio.h	Contém funções, tipos e macros de entrada e saída
stdlib.h	Contém funções para conversão de números, alocação de memória e tarefas similares
string.h	Define funções para manipulação de "strings"
time.h	Define funções e estruturas usadas na manipulação de data e hora.

Diretivas

- Diretiva **DEFINE**:
 - Mecanismo básico do C para criar macros e identificadores;
 - Precedida pelo símbolo **#** e seguida de dois argumentos:
 - Nome do identificador;
 - Conteúdo associado ao identificador.
 - `#define UFBA "Universidade Federal da Bahia"`
 - `#define AVISO "As normas preestabelecidas não permitem \`
acesso a esse módulo do sistema"
 - `#define CUBO(X) ((X)*(X)*(X))`
 - `CUBO(2)` $((2)*(2)*(2)) = 8$
 - `CUBO(A+B)` $((A+B)*(A+B)*(A+B))$
 - Funciona semelhante a uma função, mas como macro, permite que argumentos sejam suprimidos, ao contrário das funções;

Tipos Básicos de Dados

Tipos	Descrição	Tamanho ~	Faixa
char	Caractere	8 bits = 1 byte	-127 a 127
int	Inteiro	16 bits = 2 bytes	-32.767 a 32.767
float	Ponto flutuante	32 bits = 4 bytes	7 dígitos de precisão
double	Ponto flutuante de precisão dupla	64 bits = 8 bytes	10 dígitos de precisão
void	Sem valor		

Tipos de Dados Modificados

Tipo	Tamanho <i>bits</i>	Faixa
char	8	-127 a 127
unsigned char	8	0 a 255
signed char	8	-127 a 127
Int	32	-2.147.483.648 a 2.147.483.647
unsigned int	16	0 a 65.535
signed int	16	O mesmo que int
short int	16	O mesmo que int
unsigned short int	16	0 a 65.535
signed short int	16	O mesmo que short int
long int	32	-2.147.483.647 a - 2.147.483.647
signed long int	32	O mesmo que long int
unsigned long int	32	0 a 4.294.967.295
float	32	$3,4 \times 10^{-38}$ a $3,4 \times 10^{38}$
double	64	$1,7 \times 10^{-308}$ a $1,7 \times 10^{308}$
long double	80	$3,4 \times 10^{-4932}$ a $3,4 \times 10^{4932}$

Identificadores

- Nome de variáveis, funções, rótulos e outros objetos;
- Formato
 - Letra ou sublinhado (" _") seguido de letras, números ou sublinhados
 - Dependendo do tamanho, o compilador pode ignorar alguns caracteres

Correto	Incorreto
count	1count
teste23	Hi!there



Count # count # COUNT

Variáveis

- Formato
 - `tipo lista-variaveis`
- Exemplos
 - `int x, y, count;`
 - `long salary;`
 - `unsigned int id;`
 - `int x=0,y=0,z=0;`
 - `int x, y, z = 0; /* apenas z é inicializado com 0*/`

Variáveis

Globais	Locais
Reconhecidas pelo programa inteiro e podem ser usadas por qualquer parte do código	Declaradas dentro de um bloco de código: - Função, loops, controle, etc.
Criada no início do programa	É criada na entrada do bloco e destruída na saída
Alocam memória o tempo todo (Efeito Colateral)	Só podem ser referenciadas por comandos que estão dentro do bloco no qual as variáveis foram declaradas
<pre>int count; /* variável global */ int main () { count = 100; }</pre>	<pre>void func1 (){ int x; /* variável local */ x = 10; }</pre>

Exercitando...

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

int main() {

    int x;
    x = 200;
    if (x == 225){
        int count;
        count = 100;
        printf("1º valor de count %d\n",count);
        count++;
        printf("2º valor de count %d\n",count);
        count++;
    }
    printf("3º valor de count %d",count);
}
```

Conversões de Tipos

- Regra geral
 - lado direito convertido no esquerdo
 - Pode ocorrer truncamento
- Exemplo

```
int x; //4 bytes
char ch; //1 byte
float f; //4 bytes
```

```
void func () {
    ch = x;
    x = f;
    f = ch;
    f = x;
}
```

Operadores Aritméticos

Operador	Ação
-	Subtração
+	Adição
*	Multiplicação
/	Divisão
%	Resto da divisão
--	Decremento
++	Incremento



Algumas funções básicas

cin - Entrada Padrão

- `cin >> var1 >> var2 >> ... >> varN;`
- **cin** recebe uma lista de variáveis e lê valores para elas na **ordem** especificada
- Identifica automaticamente o tipo da variável e lê um valor de acordo com esse tipo
 - Variáveis reais podem ter ponto flutuante na entrada, inteiros não podem
 - Ex.:
int idade;
float peso;
cin >> idade >> peso;
- Comando lerá um número inteiro (sem ponto) para **idade** variável a e um número real (com ou sem ponto) para a variável **peso**

cout - Saída Padrão

- `cout << par1 << par2 << ... << parN;`
- `cout` recebe uma lista de parâmetros e **imprime** os mesmos na **ordem** especificada
 - Parâmetros podem ser variáveis e constantes
 - Ex.:

```
int idade=25;  
cout << "Resultado = " << idade << endl;
```
- Saída
 - "Resultado = 25\n"
 - endl representa uma quebra de linha ("\n")

scanf

- `scanf("%d %f ...", &par1, &par2, ..., &parN);`
- O padrão especifica quais dados são esperados;
- Os parâmetros devem ser fornecidos na mesma ordem em que são colocados no padrão, e indicam o destino dos valores lidos;
- No exemplo, %d refere a par1 e %f refere a par2;
- Se houver mais referências do que parâmetros, seu programa tem grandes chances de falhar;
- Deve-se usar o '&' antes dos parâmetros para indicar que estes são um destino (endereço de memória) e não um valor.

printf

- `printf("padrão \n %d \t %f ...", par1, par2, ..., parN);`

- `%d` – int;
- `%ld` – long int;
- `%f` – float;
- `%lf` – double;
- `%c` – char.

Os parâmetros devem ser fornecidos na **mesma ordem** em que são colocados no padrão.

No exemplo, `%d` refere a `par1` e `%f` refere a `par2`. Se houver mais referências do que parâmetros, seu programa tem grandes chances de falhar.

- Carcteres especiais:

- `\n` – quebra de linha;
- `\t` – tab.

- Outras opções:

- Setar o número de casa decimais:
 - `float` com **duas** casas decimais == `%.2f`;
 - `double` com **cinco** casas decimais == `%.5lf`.

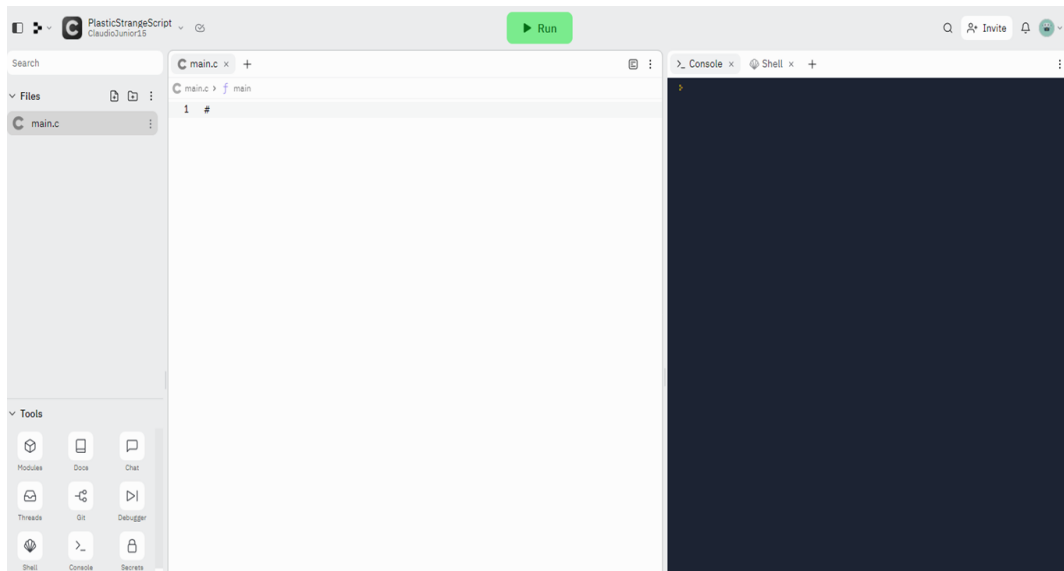


Preparando o ambiente

Online

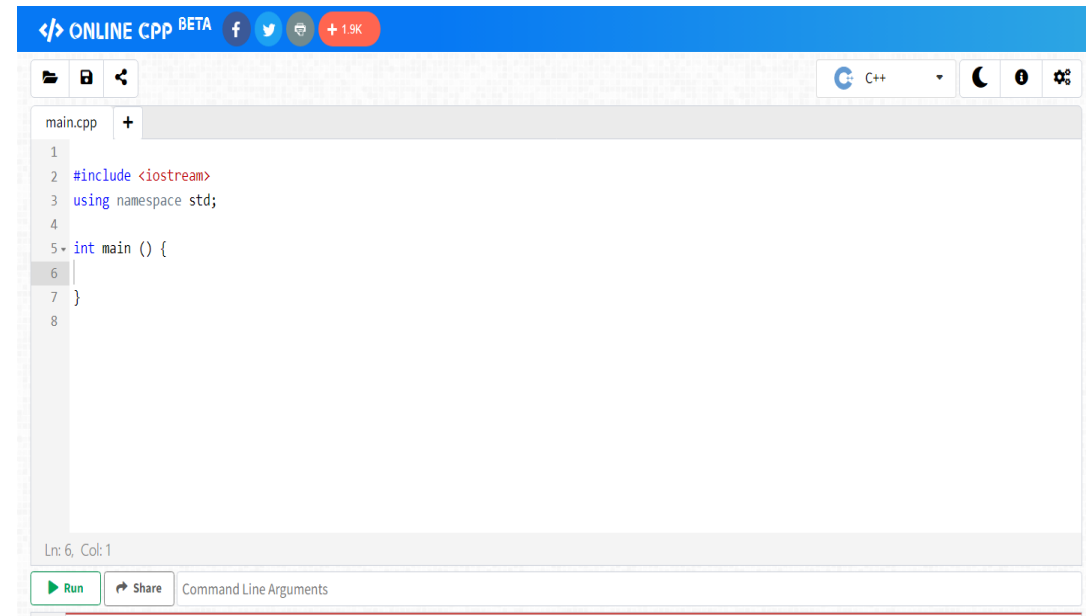
- Replit

- <https://replit.com/languages/c>



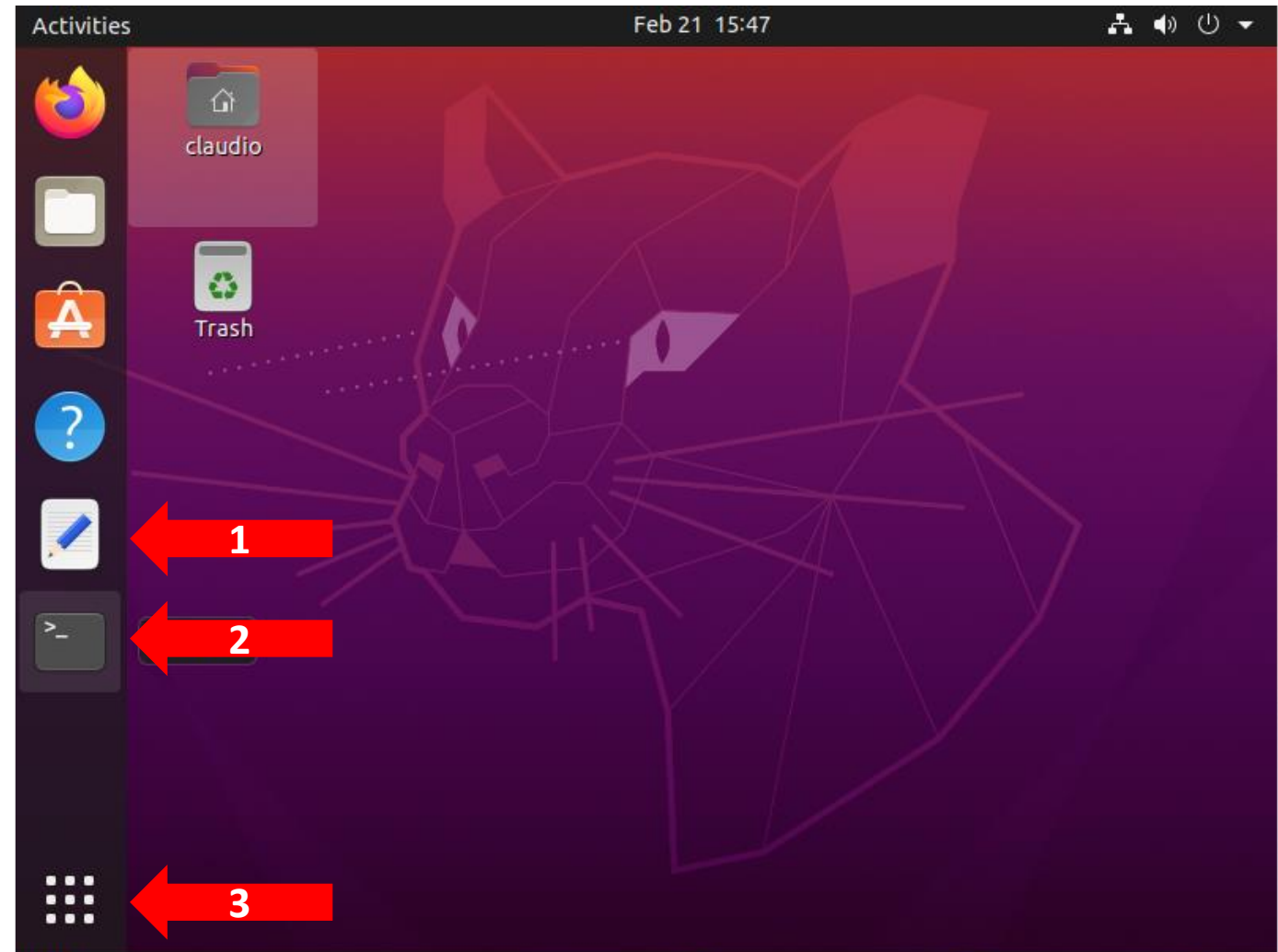
- Online-cpp

- https://www.online-cpp.com/online_c_compiler



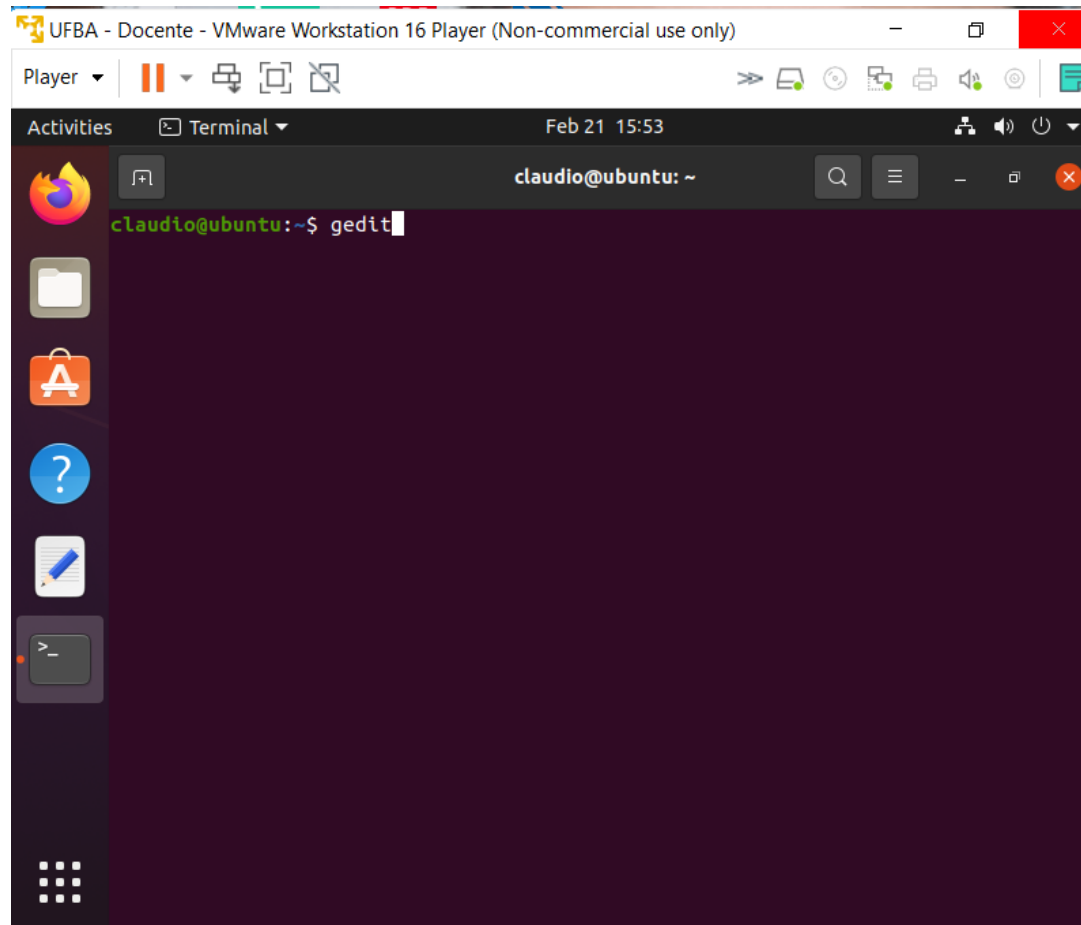
Linux

- Desktop (área de trabalho);
- Editor de Textos:
 1. Executar;
 2. Usar o Terminal;
 3. Procurar;

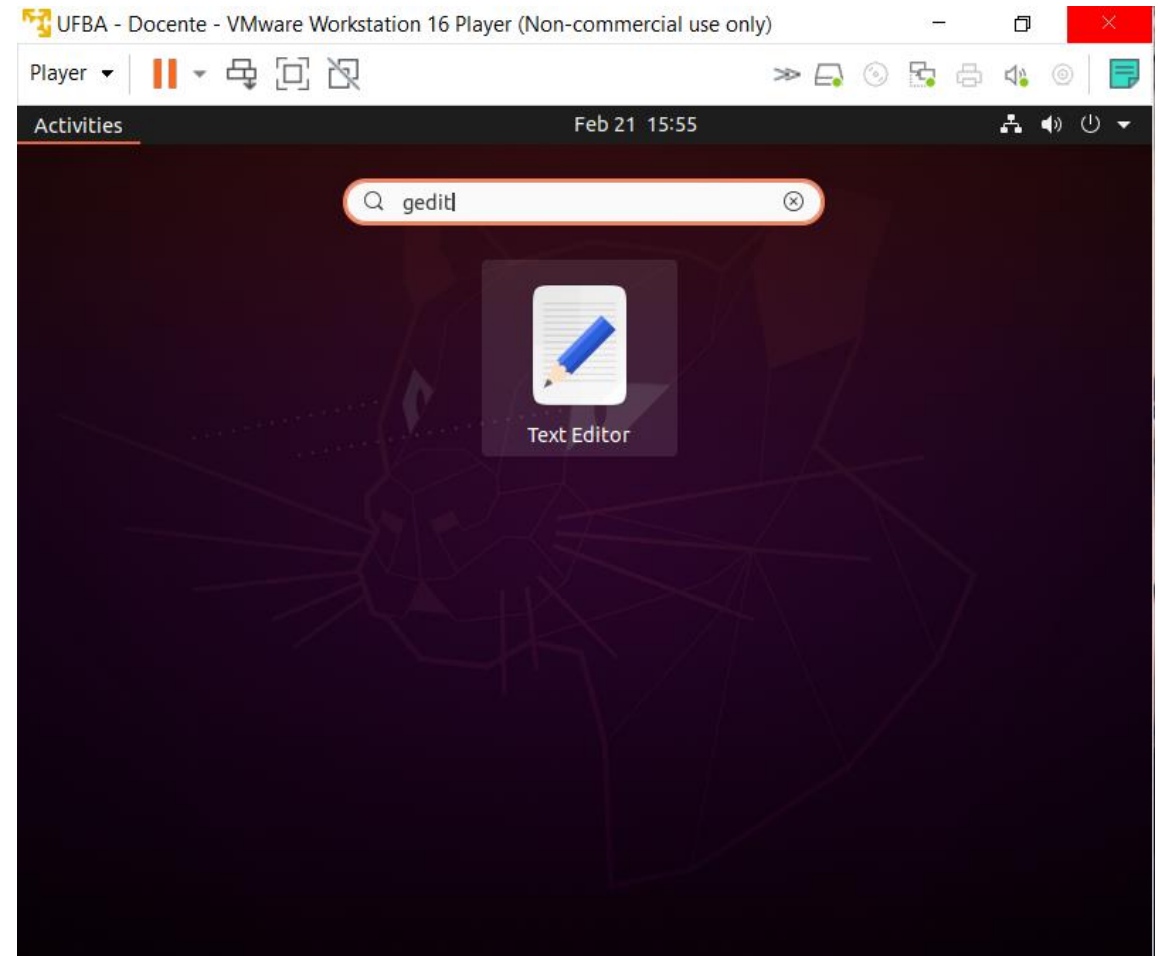


Linux

Executando o Editor pelo Terminal

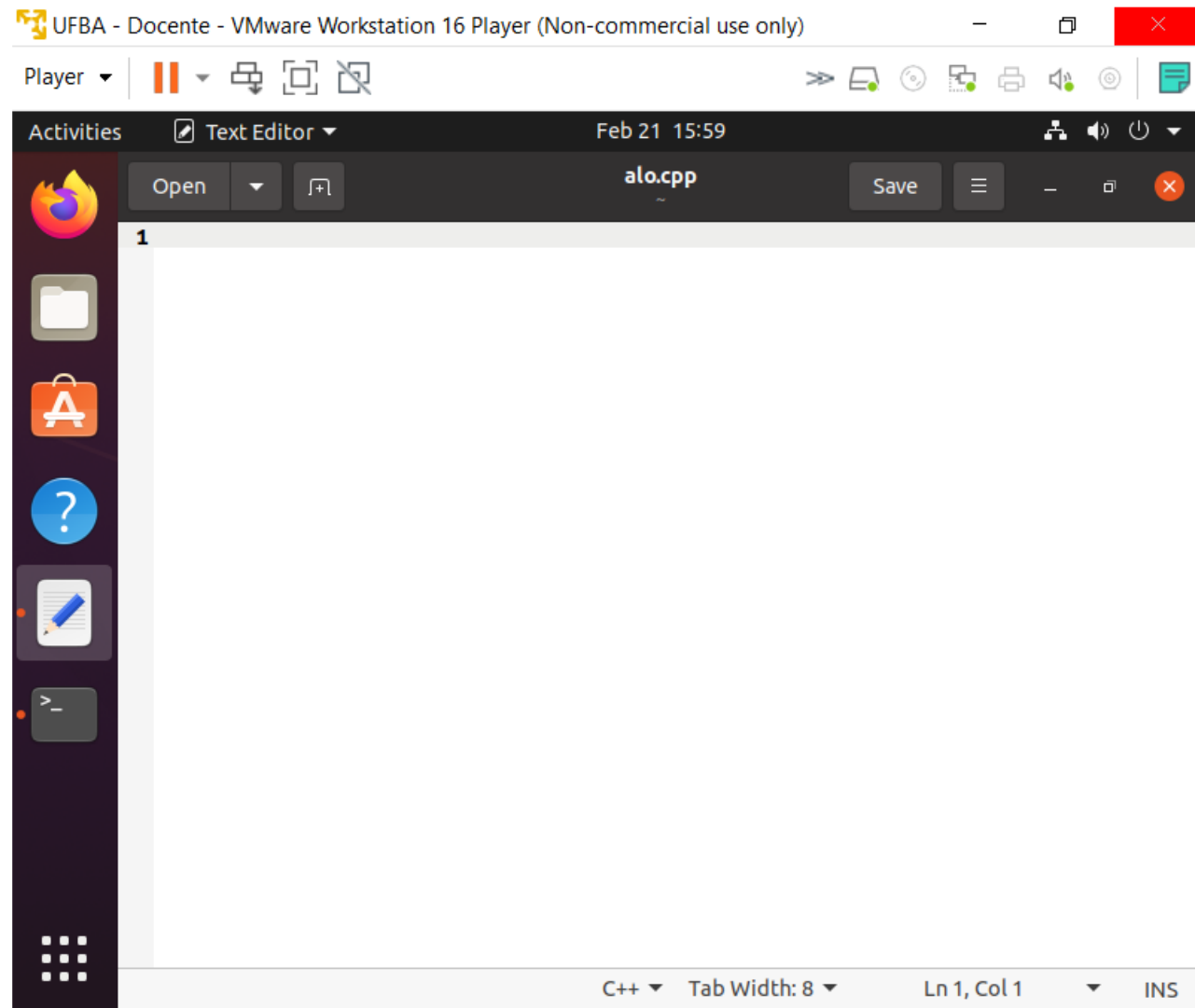


Pesquisando o Editor



Primeiros Passos

- Vamos começar...



Linux

- Para compilar :
 - `$g++ alo.cpp`
- Para executar:
 - `$/a.out`
- Compilando com G++ via linha de comando
 - Único arquivo
 - `g++ prog.cpp -o prog` (compila e gera executável)
 - `./prog` (executa o programa)
 - Diversos arquivos
 - Primeiro compile cada arquivo
 - ✓ `g++ -c prog1.cpp` (vai gerar um `prog1.o`)
 - Os vários programas-objeto devem ser montados para formar um único executável
 - ✓ `g++ -o programa prog1.o prog2.o prog3.o`

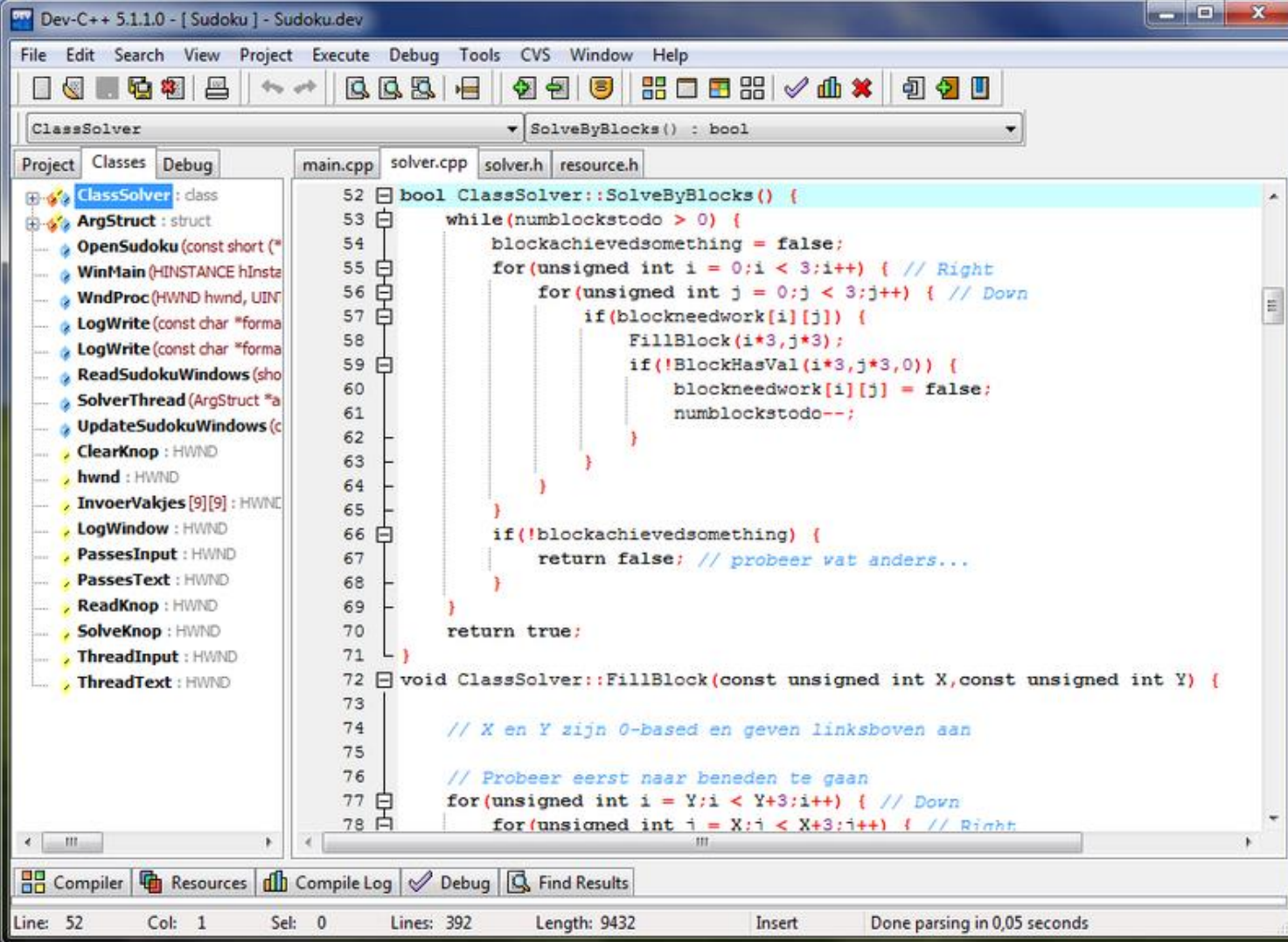
Linux

- Lista de comandos úteis:
 - **\$ cd "nome_diretório"**
 - muda o diretório corrente para o diretório "nome_diretório"
 - **\$ ls**
 - lista o conteúdo do diretório corrente
 - **\$ ls "nome_diretório"**
 - lista o conteúdo do diretório "nome_dir"
 - **\$ mkdir "nome_diretório"**
 - cria um diretório chamado "nome_diretório"
 - **\$ cat "nome_arquivo"**
 - imprime o conteúdo do arquivo "nome_arquivo" na tela

Windows

- CodeBlocks:
 - <http://www.codeblocks.org/>
 - Downloads;
 - Downloads the binary release;
 - Microsoft Windows;
 - Codeblocks-XX.Xxmingw-setup.exe;
 - Sourceforge.net.
- Dev-C++
 - <http://www.bloodshed.net/>;
 - Supports Windows 98, NT, 2000, XP;
 - Dev-C++ X.X (X.X.X.X) **with Mingw/GCC X.X.X compiler** and **GDB X.X.X debugger**.

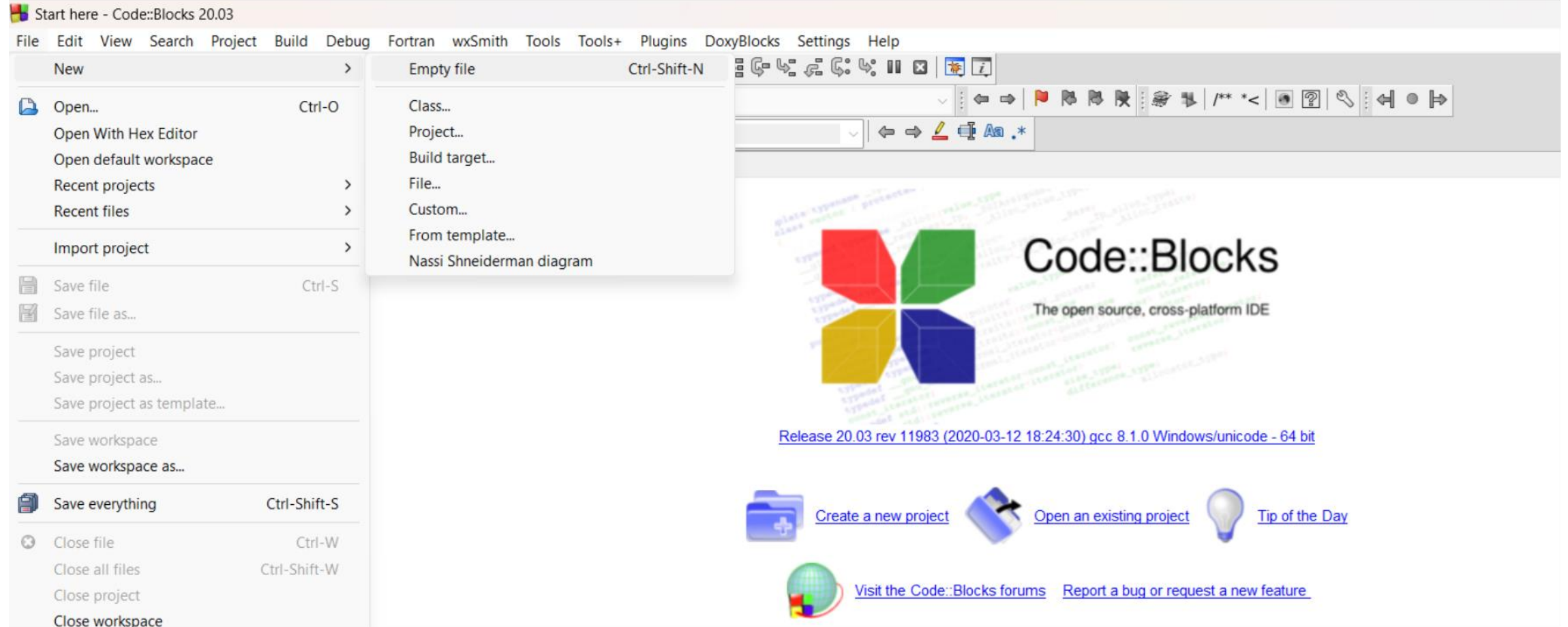
Dev-C++



```
Dev-C++ 5.11.0 - [ Sudoku ] - Sudoku.dev
File Edit Search View Project Execute Debug Tools CVS Window Help
ClassSolver SolveByBlocks() : bool
Project Classes Debug
ClassSolver : class
ArgStruct : struct
OpenSudoku (const short (*)
WinMain (HINSTANCE hInsta
WndProc (HWND hwnd, UNIN
LogWrite (const char *forma
LogWrite (const char *forma
ReadSudokuWindows (sho
SolverThread (ArgStruct *a
UpdateSudokuWindows(c
ClearKnop : HWND
hwnd : HWND
InvoerVakjes [9][9] : HWND
LogWindow : HWND
PassesInput : HWND
PassesText : HWND
ReadKnop : HWND
SolveKnop : HWND
ThreadInput : HWND
ThreadText : HWND
52 bool ClassSolver::SolveByBlocks() {
53     while(numblockstodo > 0) {
54         blockachievedsomething = false;
55         for(unsigned int i = 0; i < 3; i++) { // Right
56             for(unsigned int j = 0; j < 3; j++) { // Down
57                 if(blockneedwork[i][j]) {
58                     FillBlock(i*3, j*3);
59                     if(!BlockHasVal(i*3, j*3, 0)) {
60                         blockneedwork[i][j] = false;
61                         numblockstodo--;
62                     }
63                 }
64             }
65         }
66         if(!blockachievedsomething) {
67             return false; // probeer wat anders...
68         }
69     }
70     return true;
71 }
72 void ClassSolver::FillBlock(const unsigned int X, const unsigned int Y) {
73     // X en Y zijn 0-based en geven linksboven aan
74     // Probeer eerst naar beneden te gaan
75     for(unsigned int i = Y; i < Y+3; i++) { // Down
76         for(unsigned int j = X; j < X+3; j++) { // Right
```

Line: 52 Col: 1 Sel: 0 Lines: 392 Length: 9432 Insert Done parsing in 0,05 seconds

CodeBlocks



Praticando

- Descrição
 - Seu programa deve cumprimentar o mundo
- Entrada
 - Este programa não possui entrada
- Saída
 - Seu programa deve imprimir a sentença “**Alo mundo!**” seguido de uma quebra de linha
- Sugestão de nome do programa
 - alo.cpp

Praticando

```
#include <iostream>  
using namespace std;
```

Cabeçalho

```
int main()
```

Função principal

```
{
```

Início da Função Principal

```
    cout<<"Alo mundo!"<< endl;
```

Código

```
}
```

Fim da Função Principal

Para compilar
\$g++ alo.cpp

Para executar
\$./a.out

Processo de Compilação/Linkedição

