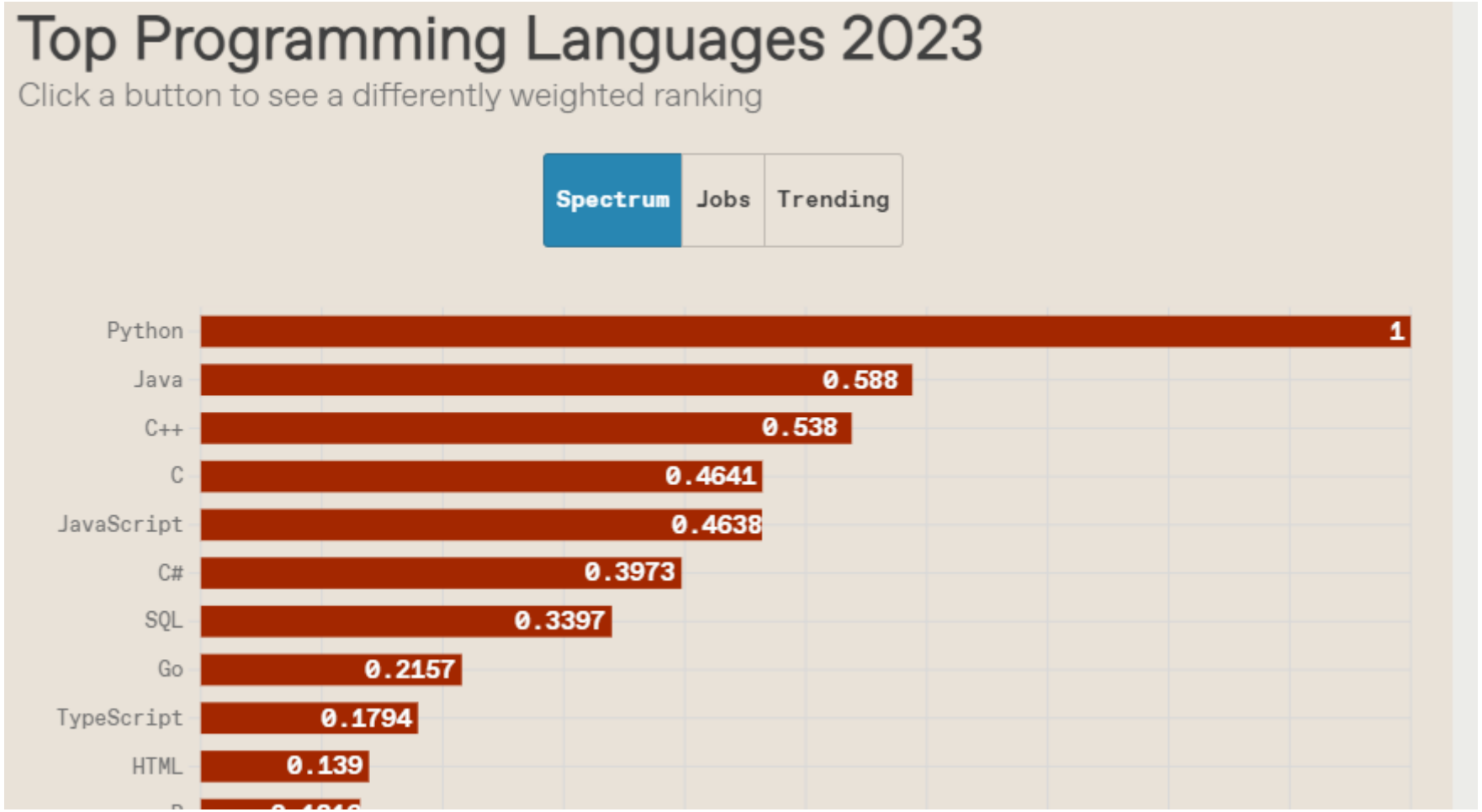
Linguagem C – 1ª parte

13/09/2023

Sumário

- C vs Java
- Compilação e execução
- Tipos pré-definidos
- Input e output formatado
- Arrays
- math.h – The mathematical library
- Ligações úteis
- Referências

IEEE Spectrum – Top Programming Languages



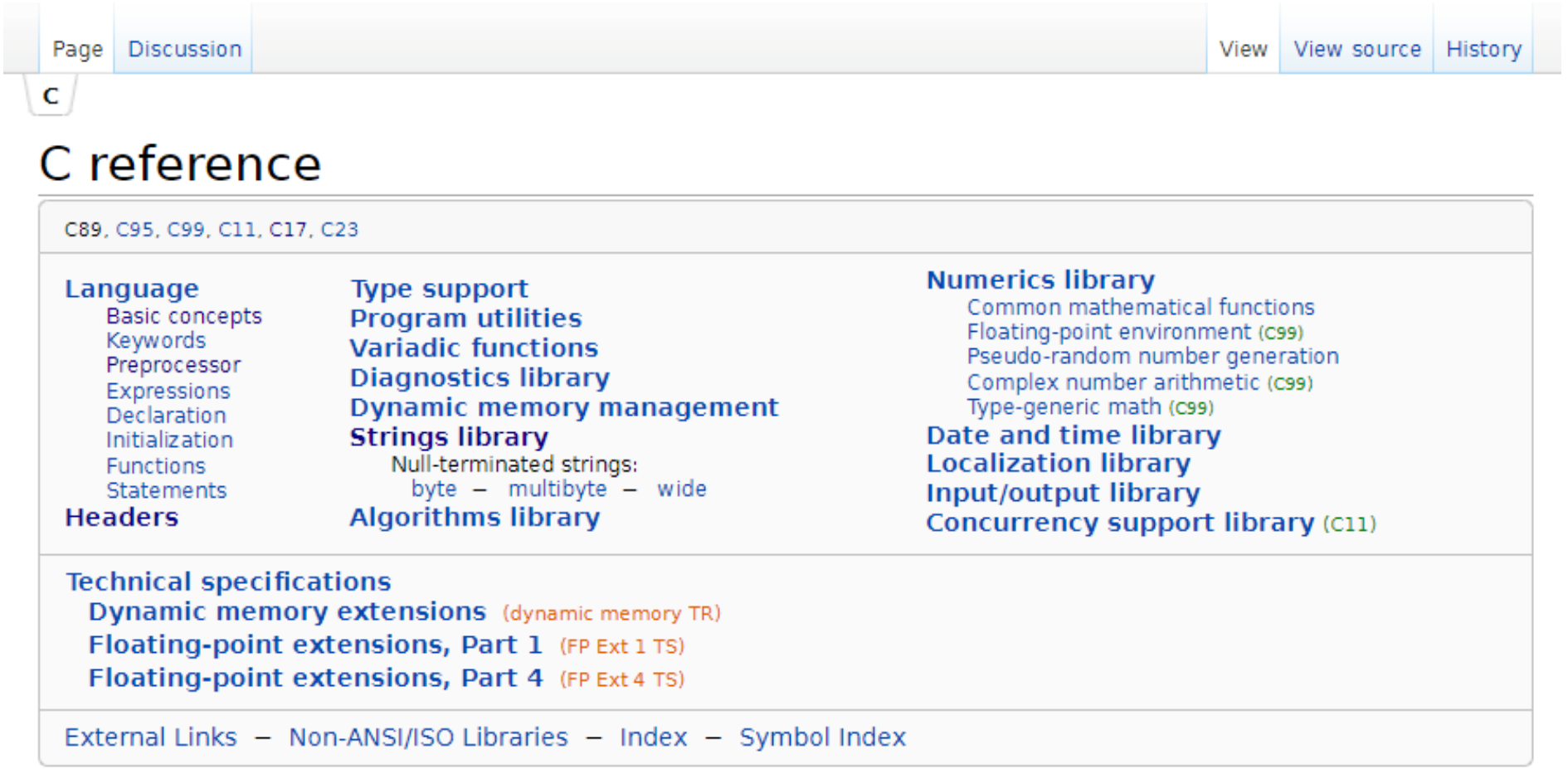
https://spectrum.ieee.org/the-top-programming-languages-2023

Porquê C ?

- Minimalista / Proximidade do hardware
  - Questões / decisões de projeto e implementação
- Rapidez
- Reduzido espaço de memória
  - Sistemas embebidos
- Interligação
  - Ligar componentes desenvolvidos em diferentes linguagens

- Semelhanças ao nível da sintaxe, mas C e Java são diferentes !!

C reference



Page | Discussion | View | View source | History

C

C reference

C89, C95, C99, C11, C17, C23

Language
- Basic concepts
- Keywords
- Preprocessor
- Expressions
- Declaration
- Initialization
- Functions
- Statements

Headers

Type support
Program utilities
Variadic functions
Diagnostics library
Dynamic memory management
Strings library
- Null-terminated strings:
  - byte – multibyte – wide

Algorithms library

Numerics library
- Common mathematical functions
- Floating-point environment (C99)
- Pseudo-random number generation
- Complex number arithmetic (C99)
- Type-generic math (C99)

Date and time library
Localization library
Input/output library
Concurrency support library (C11)

Technical specifications
- **Dynamic memory extensions** (dynamic memory TR)
- **Floating-point extensions, Part 1** (FP Ext 1 TS)
- **Floating-point extensions, Part 4** (FP Ext 4 TS)

External Links – Non-ANSI/ISO Libraries – Index – Symbol Index

https://en.cppreference.com/w/c

C vs Java – O que é "idêntico" ?

- Valores, tipos, literais, expressões
  - O tipo booleano não é um tipo pré-definido !!
  - Usar int ou char
- Variáveis
- Instruções condicionais: **if**, **switch**
- Ciclos: **while**, **for**, **do-while** – MAS, não há o iterador **for-in-collection**
- Call-return: métodos em Java, funções em C

C vs Java – O que é diferente ?

- Não há classes ou objetos !!
  - C não é uma linguagem OO, mas tem tipos estruturados: **struct** e **union**
- Possível definir tipos auxiliares: **typedef**
- Arrays são mais simples !!
  - Não conhecem o seu tamanho
  - Não é verificada a validade dos índices
- Strings não são um tipo autónomo !!
  - Sequência de caracteres, com um terminador, armazenada num array
  - C standard library

C vs Java – O que é diferente ?

- Coleções (listas, dicionários, etc.), exceções e genéricos NÃO são diretamente suportados em C

- Também não há gestão automática da memória atribuída
  - Programador tem de alocar (malloc) e de libertar (free) memória para usar estruturas de dados dinâmicas, como listas, árvores, etc.

- C permite manipular diretamente ponteiros (endereços de memória)
  - Alterar o conteúdo de uma localização de memória, dado o seu ponteiro
  - Realizar operações aritméticas sobre ponteiros; p.ex., iterar sobre um array

Programa em C

- Um **programa** é uma coleção de (uma ou mais) **funções** que processam informação armazenada em **variáveis** e **estruturas de dados** locais ou globais

- A execução de um programa inicia-se na função **main**

- Programas são **compilados** e executados diretamente no processador

hello.c

```
#include<stdio.h>

/* This is a
   comment */
int main(void) {
    // Another comment
    printf("Hello world!\n");
    return 0;
}
```

Directiva para importar o ficheiro cabeçalho que declara a função **printf** da biblioteca da linguagem

Indica que o programa terminou corretamente

Compilação e execução

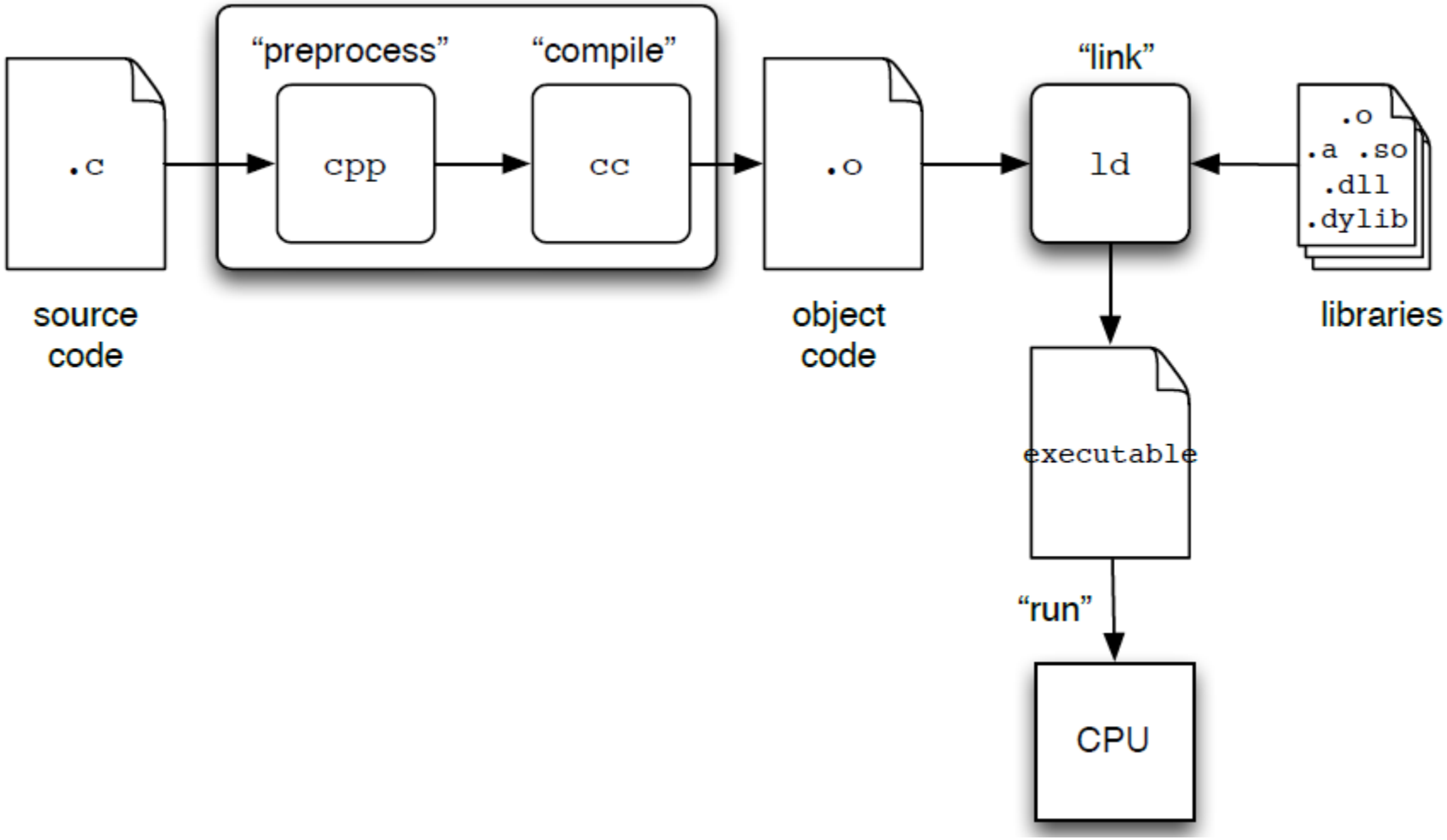
- Linux

```
cc source_file.c                    ->    ./a.out
cc -Wall source_file.c
cc -Wall my_file.c -o exec_name     ->    ./exec_name
```

- Windows

```
gcc source_file.c                   ->    .\a.exe
gcc -Wall source_file.c
gcc -Wall my_file.c -o exec_name    ->    .\exec_name
```

Compilação e execução



Tipos pré-definidos – Valores inteiros

```
            char  c0 = 'A';  // by default signed on most compilers
  signed    char  c1 = 'B';  // make sure the type is signed
unsigned    char  c2 = 'C';

            short s0 = 1763;  // the same as signed short
unsigned    short s1 = 1728;

            int   i0 = -1373762;  // the same as signed int
unsigned    int   i1 = 8382382U;  // the trailing U signals that the integer constant is unsigned

            long  l0  = 82781762873L;  // the same as signed long and signed long int
unsigned    long  l1 = 38273827322UL;  //   the int is optional, so we do usually do not put it

        long long  L0  = 8278176284398439 8473LL;  // the same as signed long long int
unsigned long long  L1 = 3827382733493498 3322ULL;  //   the int is optional
```

Tipos pré-definidos – Valores reais

```
 float f =  1.23e3f;  // the same as 1230.0f (f denotes a 4-byte floating point constant)
double d = -1.23e6;   // the same as -1230000.0 (no f, so a 8-byte floating point constant)
```

- **float**: 4 bytes, aprox. 7 casas decimais significativas
- **double**: 8 bytes, aprox. 16 casas decimais significativas
- Se nada for dito em contrário, usar double !!

stdio.h – printf – Output formatado

```
printf(formatting_string, param1, param2, …);
```

```
printf(“There are 220 students in AED\n”);
printf(“There are %d students in %s\n”, 220, “AED”);

int x = 10;
int y = 20;
printf(“%d + %d = %d\n”, x, y, x + y);
```

Alguns especificadores de formato

%% Escreve o literal %

%c Escreve um caracter

%s Escreve uma string

%d Converte um inteiro (com sinal) na sua representação decimal

%u Converte um inteiro sem sinal na sua representação decimal

%f Converte um real na sua representação decimal: [-]ddd.ddd

%e Converte um real na sua representação exponencial

stdio.h – scanf – Input formatado

```
scanf(formatting_string, &param1, &param2, …);
```

```
int my_num;
char my_char;
printf("Type a number AND a character and press enter:\n");
scanf("%d %c", &my_num, &my_char);
printf("Your number is: %d and your char is %c\n",my_num,my_char);

char first_name[30];        // Array of chars to store a string
printf("Enter your first name:\n");
scanf("%s", first_name);
printf("Hello %s\n", first_name);
```

Arrays

- Diferenças significativas para o Java, embora a sintaxe seja idêntica !!

- Em C, um array é um bloco contíguo de memória, que contém um número fixo de elementos de um mesmo tipo
- O seu tamanho não pode ser alterado em tempo de execução

```
double x_array[3];
int n_array[] = {1, 2, 3, 4};
int 2d_array[3][3];
```

#include<math.h>

- Para usar as funções matemáticas mais habituais, é necessário incluir o ficheiro cabeçalho math.h
- E usar a flag de compilação **–lm**

```
cc –Wall prog.c –o prog -lm
```

```
printf("%f\n", sqrt(16));

printf("%f\n", ceil(1.4));
printf("%f\n", floor(1.4));
```

Ligações úteis

- C reference at cppreference.com
- Learn C by examples at tutorialspoint.com
- C Tutorial at w3schools.com
- C coding tutor at pythontutor.com

Referências

George Ferguson, *C for Java Programmers*, 2017

Tomás Oliveira e Silva, *AED Lecture Notes*, 2022