

## Aula 01:

- Introdução à linguagem C
- Teste de avaliação

Prof. Jesús P. Mena-Chalco  
[jesus.mena@ufabc.edu.br](mailto:jesus.mena@ufabc.edu.br)

3Q-2017



# Linguagens de programação

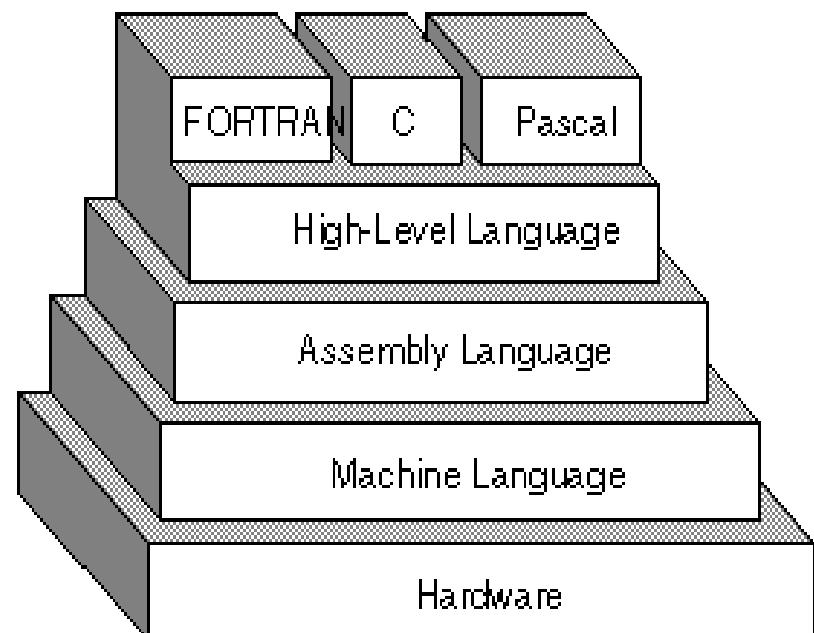
# Linguagem de programação

É um conjunto limitado de:

- **Símbolos** (comandos, identificadores, caracteres, etc)
- **Regras de sintaxe** (descrevem de forma precisa ações)

# Tipos de linguagem de programação

<b>Linguagem de máquina</b>	Compreendida pelo computador. Dependente da arquitetura do computador
<b>Linguagem de baixo nível</b>	Utiliza mnemonicos para a representação de ações elementares Ex. Assembler
<b>Linguagem de alto nível</b>	Utiliza instruções próximas da linguagem humana Ex. C, Java, Python, PHP

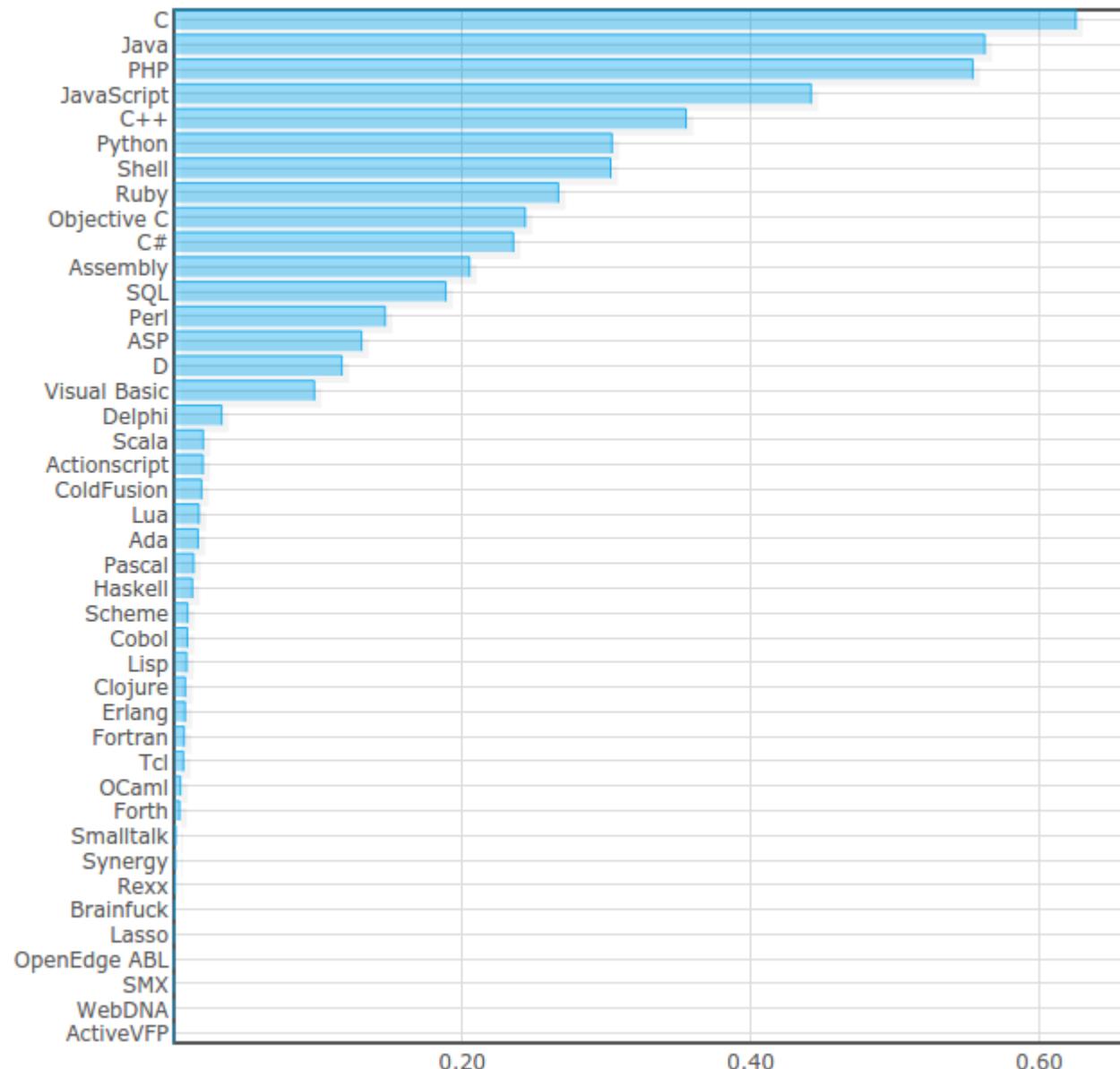


# Linguagens de programação

A dense word cloud composed of numerous programming language names, each rendered in a different color. The languages listed include:

- HPL, loke, MAD, Nu, PL/I, OPL, PHP, RPG, SAS
- CIL, Ecl, Felix, Gosu, JADE, Limbo, Mirah, NSIS, PILOT, REBOL, REXX, Slate
- Ch, CLU, Curl, Dart, EXEC, DiBOL, FoxPro, haXe, Inform, Magic
- NXT-G, OCaml, S-PLUS, Ruby, Seed7, SPSS
- Caml, CFML, COBOL, Fantom, Lua, JavaScript, Maple, MUMPS, ooc, Scratch, Suneido, TACL
- Awk, C-Shell, Ceylon, COBOL, Fortran, INTERCAL, Lisp, Nemerle, Nimrod, Scheme, Turing, Verilog
- bc, BETABlitzMax, Clojure, Cobra, Fancy, GNU-Octave, MEL, Max/MSP, Paradox, Puppet
- Smalltalk, thinBasic, Xen
- Ada, Basic, AutoLISP, Clipper, Emacs-Lisp, Fancy, Informix-4GL, Logtalk, OpenCL, POV-Ray, Squirrel, VBScript, Shell
- ASP, Angelscript, C-Omega, ECMA Script, JScript.NET, ML, MAXScript, PostScript, VimL, Standard-ML, XBase++, Transact-SQL, Z-shell
- AutoIt, AppleScript, Bro, CoffeeScript, Euphoria, Ladder-Logic, Objective-C, Processing, Pike, Visual-Basic-.NET
- Alice, ActionScript, Dylan, Common-Lisp, Icon, Java-FX-Script, Opa, Occam, PowerScript, Smarty, XQuery, XPL
- Algol, ABAP, 4th-Dimension/4D, BCPL, Delphi/Object-Pascal, M4, Mathematica, OpenEdge-ABL, SQR, SuperCollider, Yorick
- ABC, AspectJ, C++, CL, DCPU-16-ASM, Google-Apps, Script, NQC, Object-Rexx, PowerBasic, Tex, Whitespace, yacc
- ATLAS, AutoHotkey, Bourne-Shell, Etoys, Groovy, JavaScript, MS-DOS-Batch, Revolution, Tcl, TypeScript, xBase
- API, Agilent-VEE, Assembly, Go, ColdFusion, IDL, HyperTalk, Malbolge, Objective-J, REALBasic, Vala/Genie, XSLT
- Arc, Apex, Automator, Factor, Erlang, Gambas, Korn-Shell, Mercury, sed, Pure-Data, Squeak, WebDNA
- Augeas, C++/CLI, Clarion, Fortress, LabVIEW, Modula-3, NATURAL, Python, Simulink, VHDL, X10
- Bash, Avenue, Boo, COMAL, Coq, Haskell, MATLAB, Modula-2, PL/SQL, Prolog, SIGNAL, SPARK, TOM
- cg, CHILL, Clean, Forth, Falcon, Logo, Magik, MANTIS, Oberon, Racket, Sather, Scala
- cT, DCL, Elixir, Eiffel, Heron, Kotlin, Lustre, Lasso, Moto, MOO, Pliant, Rust, Self
- eC, EGL, Julia, Java, LPC, Parrot, Pascal, Scilab, Simula
- Miva, Oz, Perl

# Linguagens de programação



(\*) Popularidade das LPs <http://langpop.com/>

# Sobre a linguagem de programação

- Atualmente existem várias linguagens que são consideradas para este tipo de disciplinas...  
(Python, C, C++, Java, Haskell, Ruby)
- Também vários paradigmas de programação (e.g. procedural, orientado a objetos,) podem ser consideradas...

**Todo programador competente deve saber/entender a linguagem C/C++.**

Tradicionalmente é utilizada a linguagem C.

**Nessa disciplina usaremos C.**



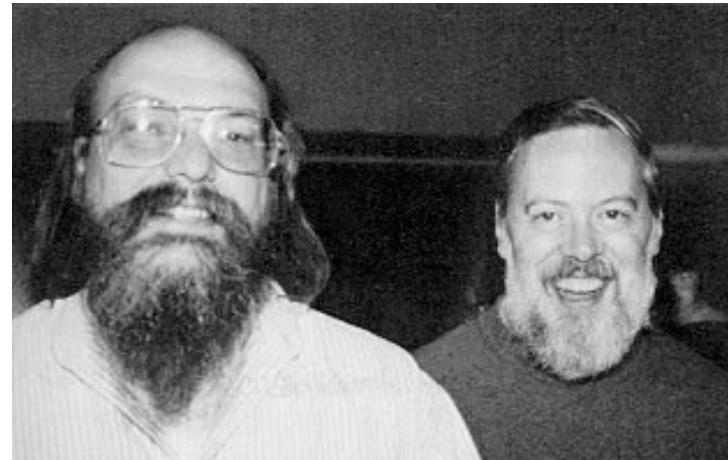
# A linguagem de programação C

# Sobre a linguagem de programação C

- Permite escrever programas de alta performance.
- C funciona em um nível mais baixo que outras linguagens (da uma ideia melhor do que realmente está acontecendo).
- É preferida no mundo científico:
  - Poucas palavras reservadas.
  - Bom controle da máquina.
  - SO livres (Linux/UNIX) são feitos em C.
  - Base de outras linguagens: C++, Java, C#

# Sobre a linguagem de programação C

- C é uma linguagem estruturada, desenvolvida nos Laboratórios BELL (1969-1972), por Dennis Ritchie.



Kenneth L. Thompson (ling. B)  
Dennis M. Ritchie (ling. C)

- **Dialectos:**
  - K&R C (1978)
  - ANSI C
  - ISO C
  - C99
  - C11 (Dez. 2011)

# The National Medal of Technology and Innovation 1998 Laureates

Kenneth L. Thompson (1943-)

Dennis M. Ritchie (1941-2011)



# Sobre a linguagem de programação C

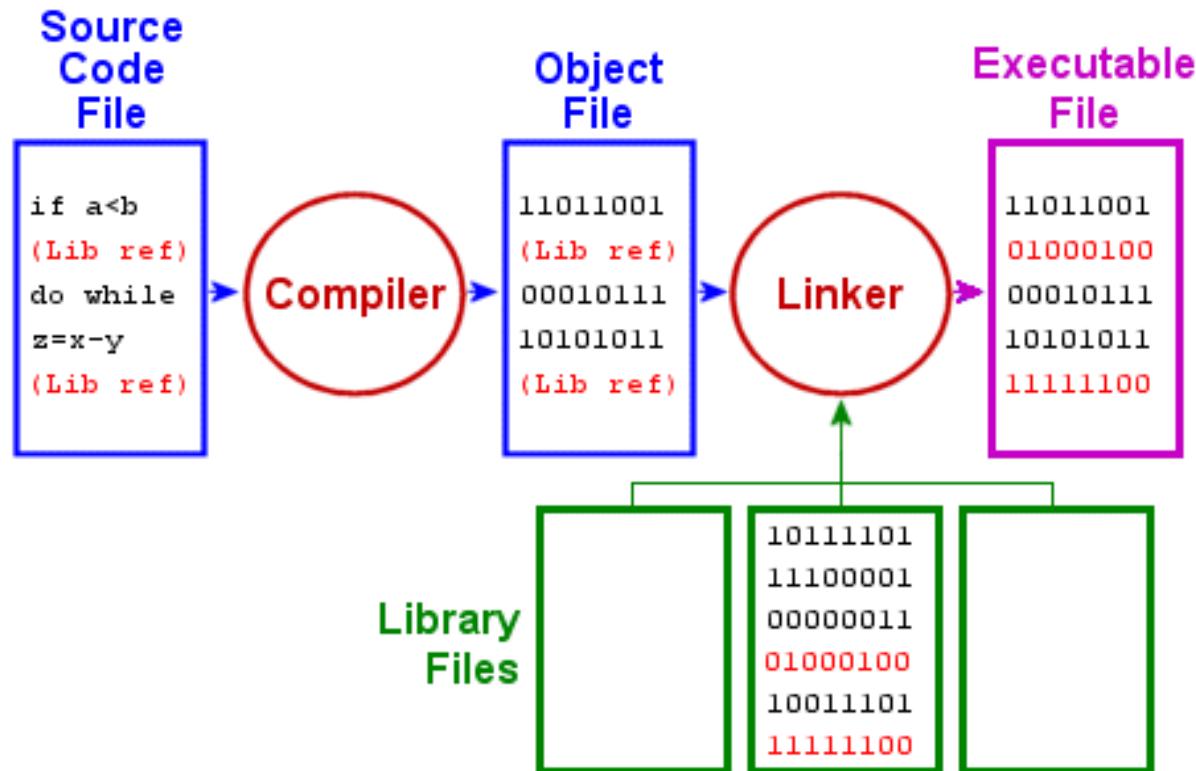
- **Influenciada por:**

- ALGOL 68
- Assembly
- B
- BCPL
- CPL
- Fortran
- PL/I

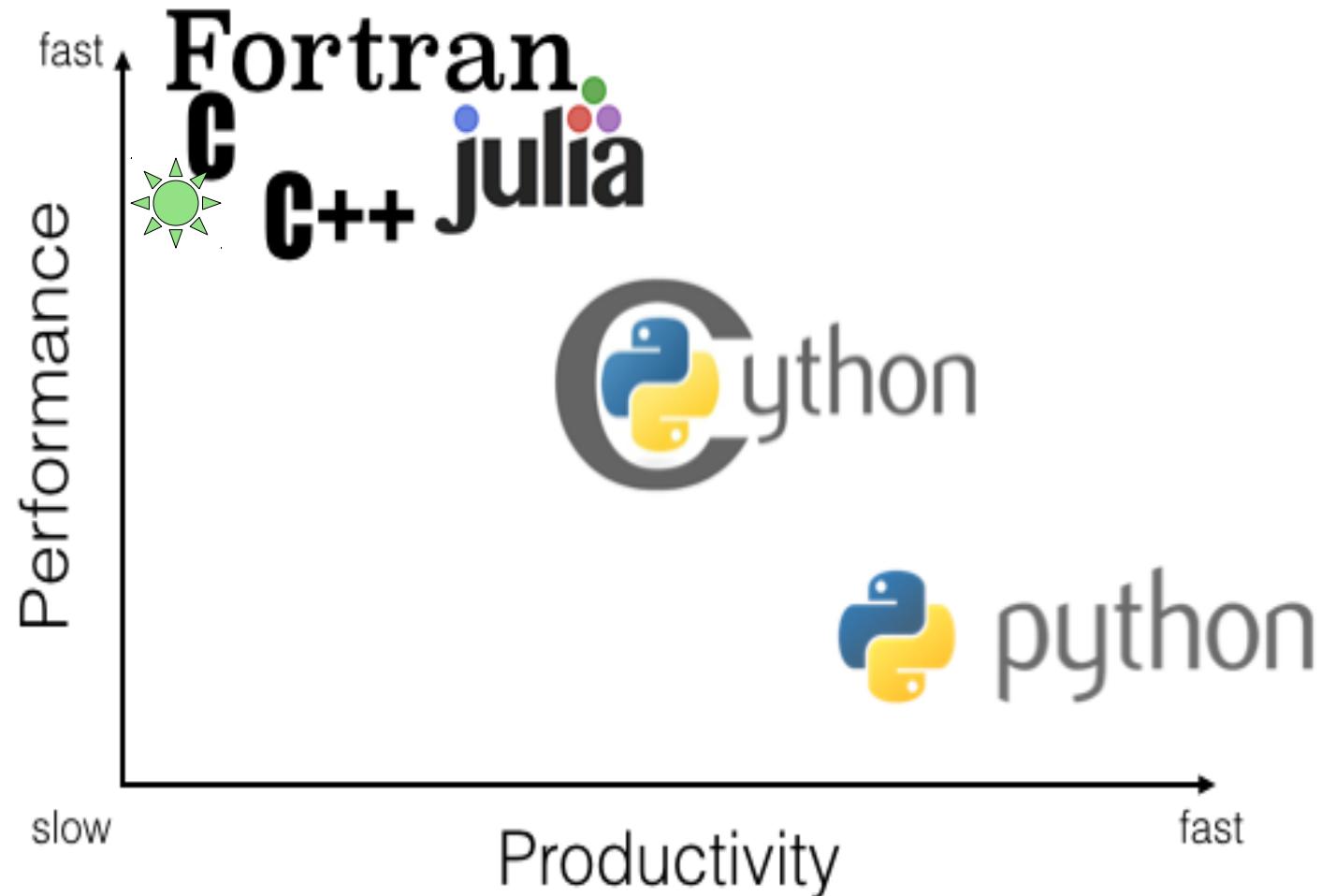
- **Influenciou:**

- AWK, BitC, C++, C#, C Shell, D, Euphoria, Go, Java, JavaScript, Limbo, Logic Basic, Objective-C, Perl, PHP, Python, ...

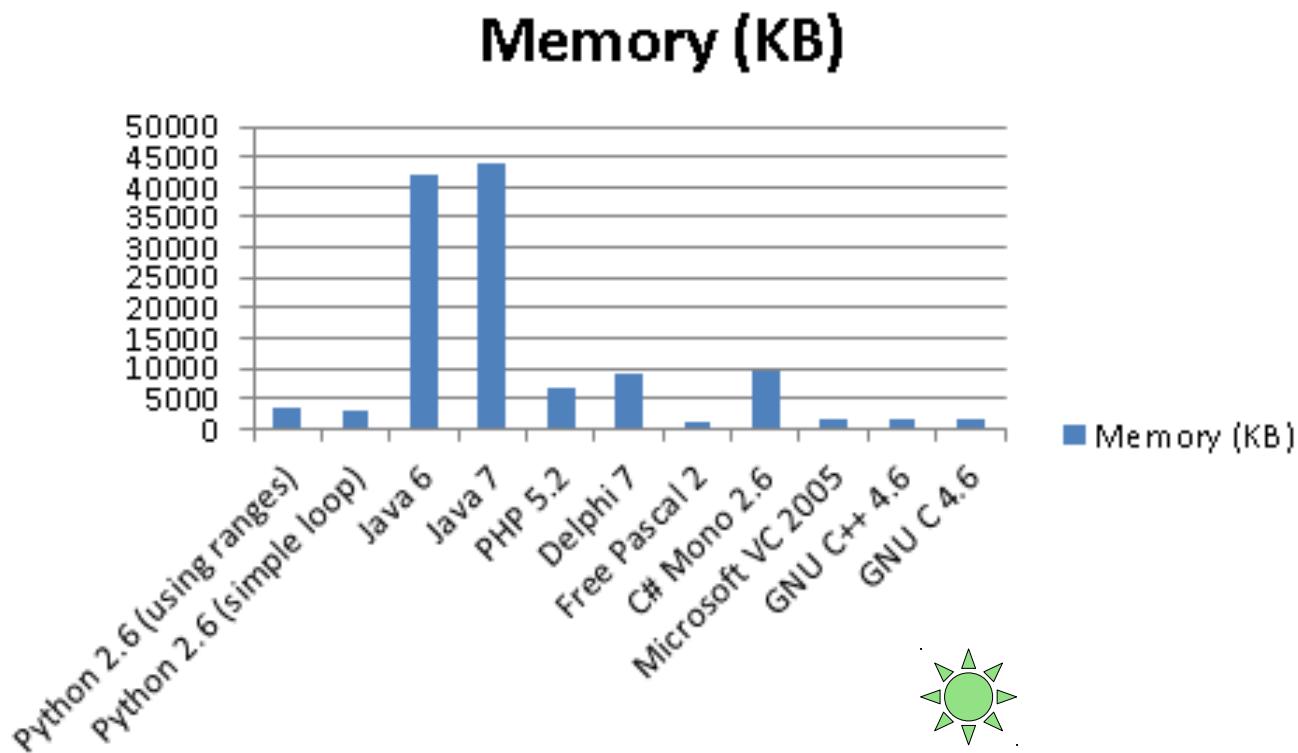
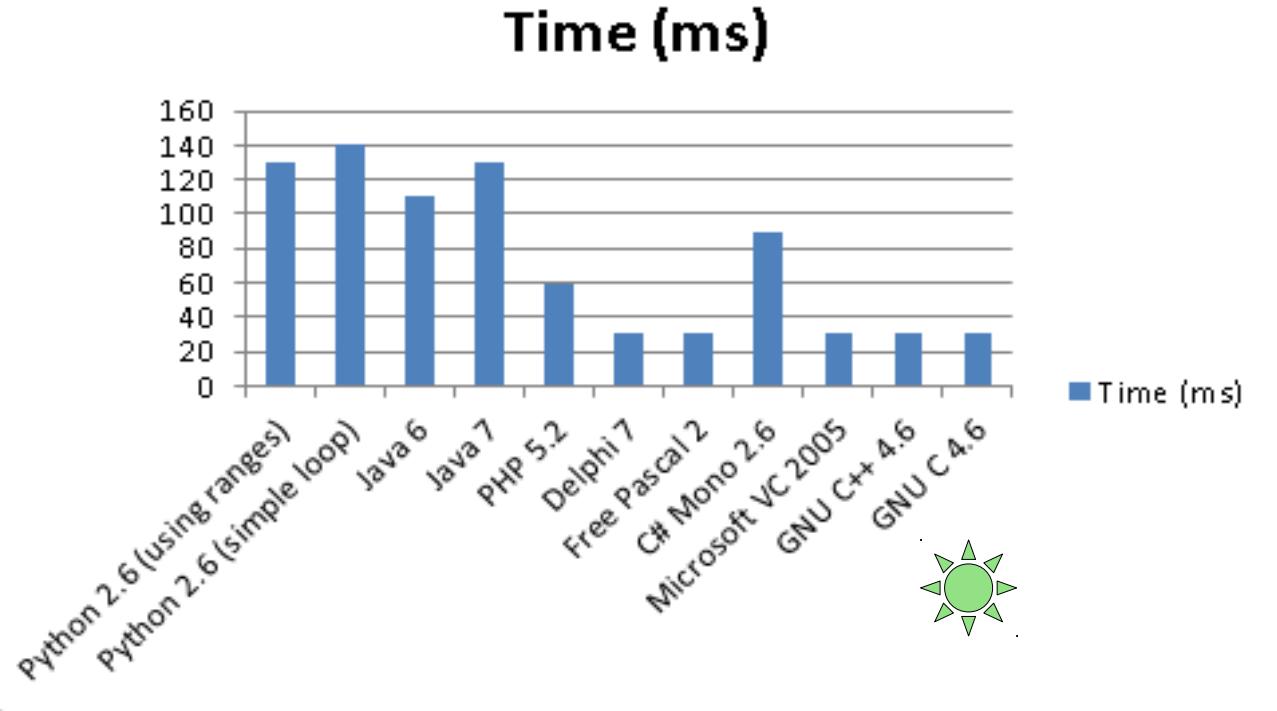
# Sobre a linguagem de programação C



- Não possui suporte a orientação a objetos
- Linguagem de nível intermediário:
  - controle mais direto do hardware, porém
  - também suporta estruturas complexas
- Gerenciamento de memória explícito
- Detecção de erro explícita (sem try/catch)
- Maior performance do programa final
- Maior dificuldade de manutenção



# Comparação subjetiva



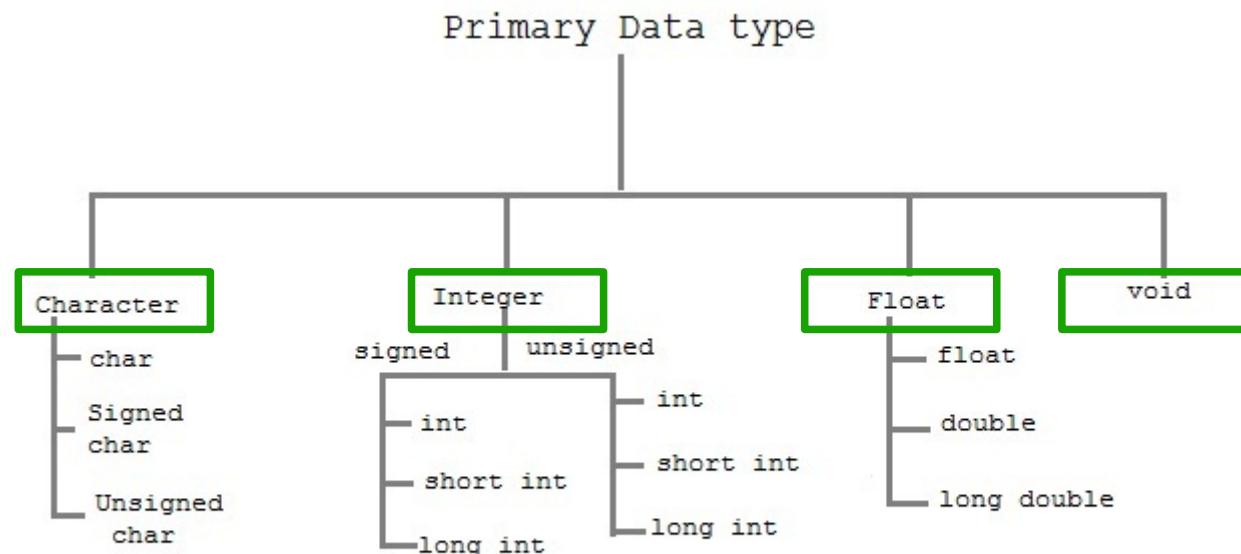
# Linguagem C

- char tem 8 bits (não é 16 bits como em Java)
- não existe tipo booleano (usar int ou char):
  - 0 falso
  - ≠ 0 verdadeiro

Na verdade C99 permite o uso de palavras true/false nos programas (mas o compilador os trata como os valores 1 e 0 de qualquer forma)
- tipos inteiros podem ser **signed** ou **unsigned**
- não tem tipo string: usa-se vetor de char

# Linguagem C: Tipos de dados

- **Tipos de dados primários.**
- Tipos de dados derivados.
- Tipos definidos pelo usuário.



# Linguagem C: Números inteiros

## Size and range of Integer type on 16-bit machine

Type	Size(bytes)	Range
int or signed int	2	-32,768 to 32767
unsigned int	2	0 to 65535
short int or signed short int	1	-128 to 127
long int or signed long int	4	-2,147,483,648 to 2,147,483,647
unsigned long int	4	0 to 4,294,967,295

# Linguagem C: Números reais

## Size and range of Integer type on 16-bit machine

Type	Size(bytes)	Range
Float	4	3.4E-38 to 3.4E+38
double	8	1.7E-308 to 1.7E+308
long double	10	3.4E-4932 to 1.1E+4932

# Linguagem C: Caracteres

**Size and range of Integer type on 16-bit machine**

Type	Size(bytes)	Range
char or signed char	1	-128 to 127
unsigned char	1	0 to 255

# Linguagem C: void

## void type

void type means no value. This is usually used to specify the type of functions.

# Tipos de dados

```
1 #include<stdio.h>
2
3 int main()
4 {
5     printf("Size of char is      %ld bytes\n", sizeof(char));
6     printf("Size of short is     %ld bytes\n", sizeof(short));
7     printf("Size of int is       %ld bytes\n", sizeof(int));
8     printf("Size of long is      %ld bytes\n", sizeof(long));
9     printf("Size of float is     %ld bytes\n", sizeof(float));
10    printf("Size of double is    %ld bytes\n", sizeof(double));
11    printf("Size of long double is %ld bytes\n", sizeof(long double));
12    return 0;
13 }
```

```
Size of char is      1 bytes
Size of short is     2 bytes
Size of int is       4 bytes
Size of long is      8 bytes
Size of float is     4 bytes
Size of double is    8 bytes
Size of long double is 16 bytes
```

```
$ uname -a
Linux xxxxxxxxx 4.2.0-c9 #1 SMP x86_64 x86_64 x86_64 GNU/Linux
```

# Conversão de tipo

```
int a = 20;  
long b;  
  
b = (long) a;
```

→ Valor preservado, só o tipo é alterado

# Conversão de tipo

```
int a;  
long b = 32;
```

```
a = (int) b;
```

→ Perigoso:  
se **b** não “cabe” em um **int** o valor será truncado

# Conversão de tipo

```
int a = 17,  
int b = 3;  
  
double x = a / b;
```

→ Neste caso x contém o valor 5

# Conversão de tipo

```
int a = 17,  
int b = 3;
```

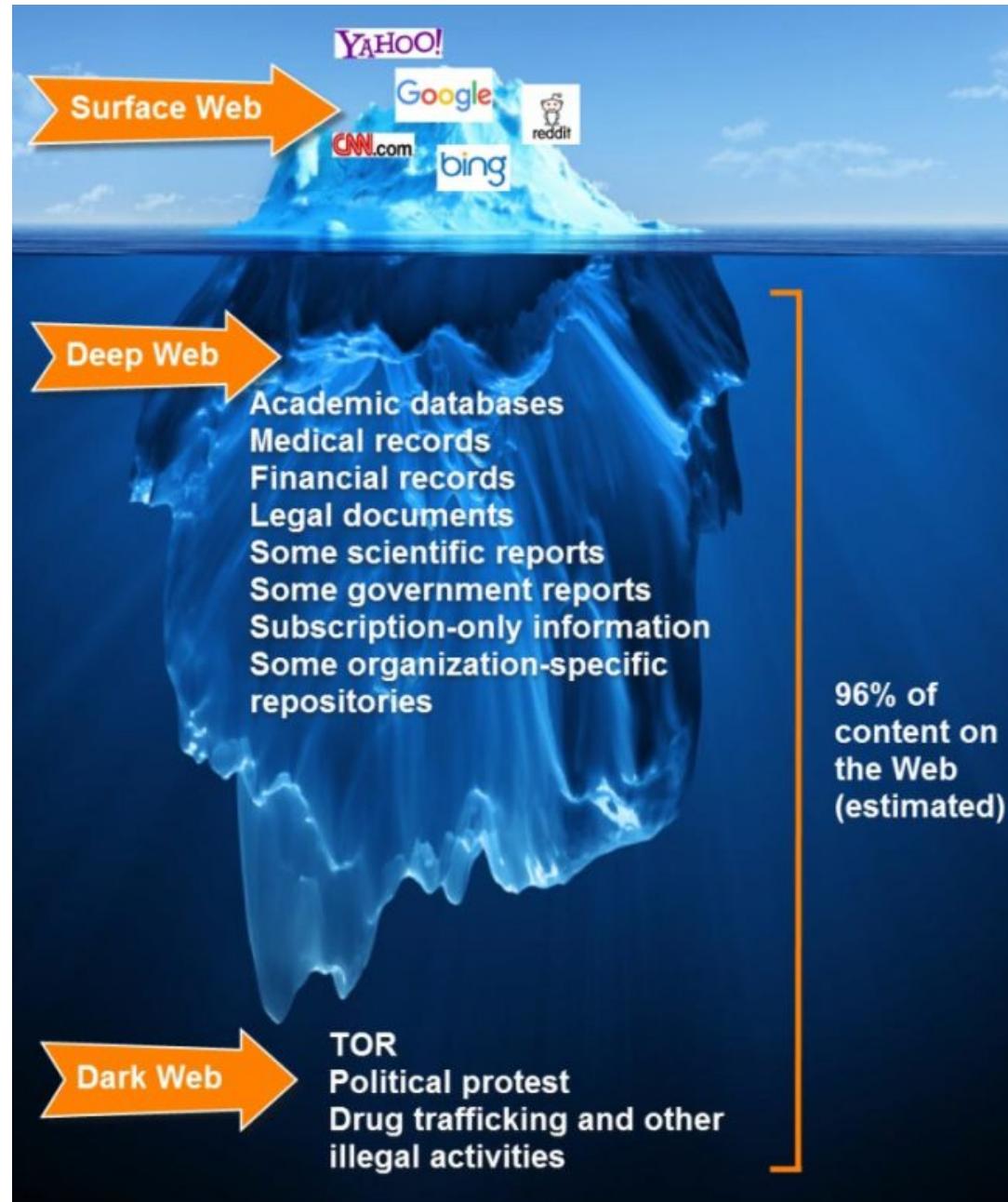
```
double x = a / (double) b;
```

→ Neste caso x contém o valor 5.66667

# Conversão de tipo

```
double x = 5.93487;  
int j = (int) x;
```

→ Neste caso j contém o valor 5





# **Teste de avaliação 01**

# Operadores unários ++ e --

- Diferentes linguagens tem operadores que permitem o incremento ou decremento de valores de forma **abreviada**.

```
1 #include <stdio.h>
2
3 void main() {
4     int A[10] = {6,7,8,9,0,1,2,3,4,5};
5     int max = A[0];
6     int i;
7
8     for (i=1; i<10; i++) {
9         if (max < A[i]) {
10             max = A[i];
11         }
12     }
13
14     printf("Valor maximo: %d\n", max);
15 }
```

```
1 #include <stdio.h>
2
3 void main() {
4     int A[10] = {6,7,8,9,0,1,2,3,4,5};
5     int max = A[0];
6     int i;
7
8     for (i=1; i<10; ++i) {
9         if (max < A[i]) {
10             max = A[i];
11         }
12     }
13
14     printf("Valor maximo: %d\n", max);
15 }
```

# Operadores unários ++ e --

- Incremento: ++
- Decremento: --

## Exemplos

$i = i + 1 \rightarrow i++$

$i = i + 1 \rightarrow ++i$

$i = i - 1 \rightarrow i--$

$i = i - 1 \rightarrow --i$

# Operadores unários ++ e --

- int i = 10;
  - p = i++
- 
- int i = 10;
  - p = ++i

# Operadores unários ++ e --

- int i = 10;
- p = i++

p=10  
i=11

- int i = 10;
- p = ++i

p=11  
i=11

**Quando o operador está antes da variável, esta é operada antes de ser utilizada.**

# Operadores unários ++ e --

- Nunca use esses operadores em variáveis que apareçam mais de uma vez na instrução.

```
1 #include <stdio.h>
2
3 void main() {
4     int x = 9;
5
6     printf("%d\n", x++ + ++x);
7     printf("%d\n", ++x + x++);
8 }
```

# Atribuição composta com operadores

## Exemplos

$i = i + 3 \rightarrow i += 3$

$i = i - 3 \rightarrow i -= 3$

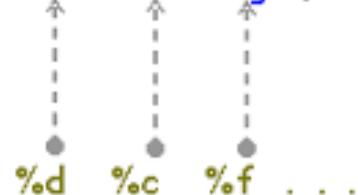
$i = i * (3 + q) \rightarrow i *= 3 + q$

$i = i / 67 \rightarrow i /= 67$

$i = i \% 10 \rightarrow i \% = 10$

## Forma geral do printf:

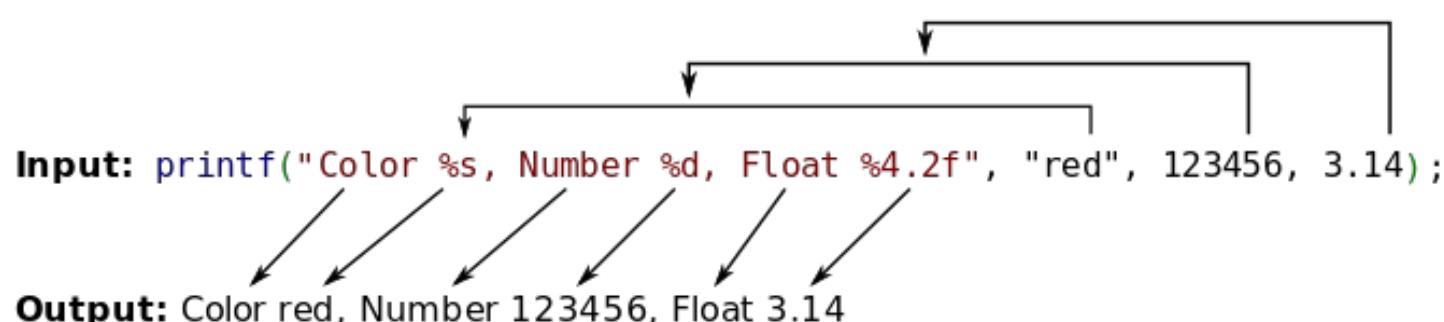
```
printf(" format string ", list of variables)
```



Formatting characters

## Especificadores se formato:

%d	%i	Decimal signed integer.
%o		Octal integer.
%x	%X	Hex integer.
%u		Unsigned integer.
%c		Character.
%s		String. See below.
%f		double
%e	%E	double.
%g	%G	double.
%p		pointer.





# **Teste de avaliação 01**

# Questão 1

```
1 #include "stdio.h"
2
3 void main () {
4     int x;
5     x = 3;
6     printf("%d", x*x);
7 }
```

O programa ao lado:

- (a) Imprime "%d"
- (b) Imprime "%3"
- (c) Imprime 3
- (d) Imprime "3\*3"
- (e) Imprime 9**

# Questão 2

```
1 #include "stdio.h"
2
3 void main () {
4     int x;
5     x = 3;
6     printf("%d %d", x, x);
7 }
```

O programa ao lado:

- (a) Imprime “%d %d”
- (b) Imprime “3 3”**
- (c) Imprime 3
- (d) Imprime “3\*3”
- (e) Imprime 9

# Questão 3

```
1 #include "stdio.h"
2
3 void main () {
4     int i, acc=0;
5
6     for(i=1; i<20; i++) {
7         acc += 2;
8     }
9     printf("%d", acc);
10 }
```

O programa ao lado:

- (a) imprime 2
- (b) Imprime 20
- (c) Imprime 38**
- (d) Imprime 40
- (e) Imprime 42

# Questão 4

```
1 #include "stdio.h"
2
3 void main () {
4     int i, acc=0;
5
6     for(i=1; i<20; i++) {
7         acc += 2;
8     }
9     printf("%d", i);
10 }
```

O programa ao lado:

- (a) imprime 2
- (b) Imprime 20**
- (c) Imprime 38
- (d) Imprime 40
- (e) Imprime 42

# Questão 5

```
1 #include "stdio.h"
2
3 void main () {
4     int i, acc=0;
5
6     for(i=0; i<2017; i++) {
7         acc += i;
8     }
9     printf("%d", acc);
10 }
```

O programa ao lado:

- (a) imprime 2031120
- (b) Imprime 2031120
- (c) Imprime 2032128
- (d) Imprime 2033136**
- (e) Imprime 2035153

# Questão 6

```
1 #include "stdio.h"
2
3 void main () {
4     int i, j, acc=0;
5
6     for(i=0; i<10; i++) {
7         for(j=0; j<i; j++) {
8             acc -= 1;
9         }
10    }
11    printf("%d", acc);
12 }
```

O programa ao lado:

- (a) Imprime -10
- (b) Imprime -35
- (c) Imprime -40
- (d) Imprime -45**
- (e) Imprime -55

# Questão 7

```
1 #include "stdio.h"
2
3 void main () {
4     int i, j, acc=0;
5
6     for(i=0; i<10; i++) {
7         acc -= 1;
8         for(j=0; j<i; j++) {
9             acc -= 1;
10        }
11    }
12    printf("%d", acc);
13 }
```

O programa ao lado:

- (a) Imprime -10
- (b) Imprime -35
- (c) Imprime -40
- (d) Imprime -45
- (e) Imprime -55**

# Questão 8

```
1 #include "stdio.h"
2
3 void main () {
4     printf("%d", 3+4*5/6*7-1);
5 }
```

O programa ao lado:

- (a) Imprime 2
- (b) Imprime 2.476199
- (c) Imprime 23**
- (d) Imprime 25.333333
- (e) NA

# Questão 9

```
1 #include "stdio.h"
2
3 void main () {
4     printf("%.2f", 3.141526);
5 }
```

O programa ao lado:

- (a) Imprime 3.141526
- (b) Imprime 3.141526.2
- (c) Imprime 3.14**
- (d) Imprime 3.2
- (e) NA

# Questão 10

```
1 #include "stdio.h"
2
3 void main () {
4     int p=10;
5     if (p==10)
6         p += 2;
7     if (p==12)
8         p = 5;
9     else
10        p = 10;
11     printf("%d", p);
12 }
```

O programa ao lado:

- (a) Imprime 15
- (b) Imprime 12
- (c) Imprime 10
- (d) Imprime 5**
- (e) NA

# Desafio

```
int Funcao1 (int N) {
    int n, i, sum=0;
    for (n=N; n>0; n=n/2)
        for (i=0; i<n; i++)
            sum++;
    return sum;
}
```

