

Programação e Desenvolvimento de Software I

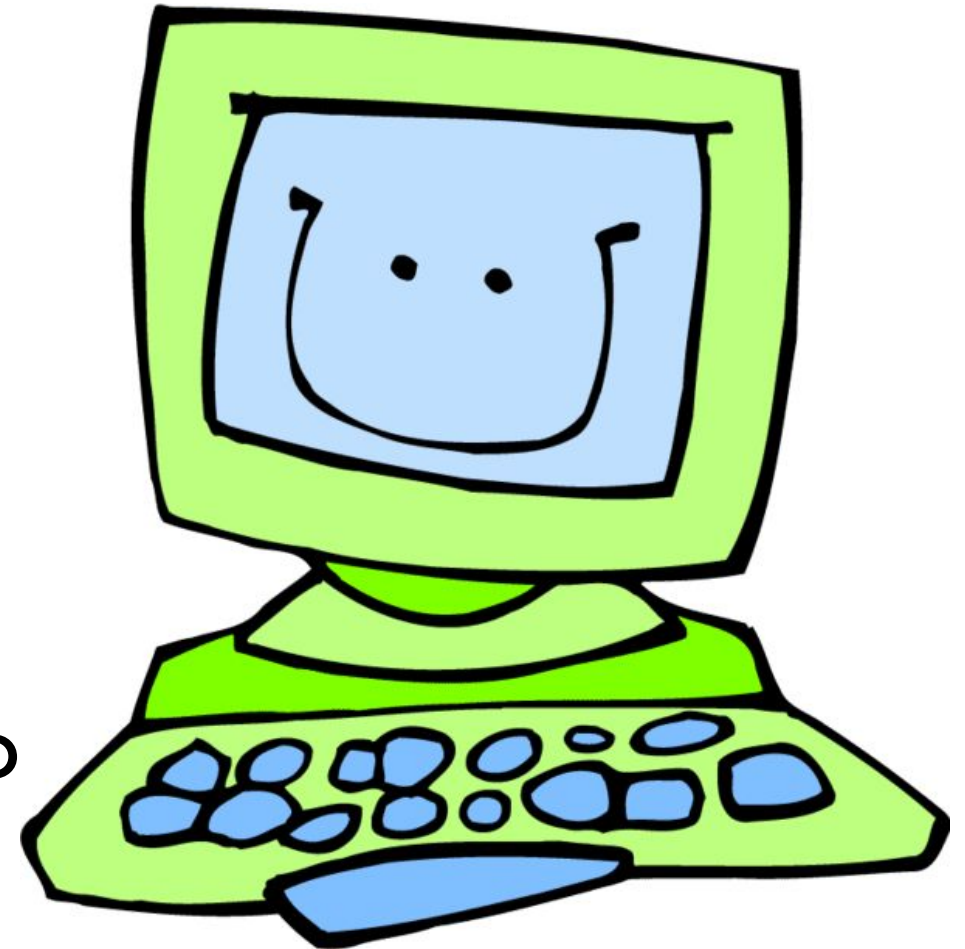
Compilação, memória e variáveis

Prof. Héctor Azpúrua
(slides adaptados do Prof. Pedro Olmo)

Introdução

Computadores

- Por que usar um computador?
 - Permite que realizemos contas matemáticas muito rapidamente!
 - Capacidade de armazenamento:
 - Videos, musicas, livros, jogos...
 - Comunicação!
 - Uso da internet, redes sociais...
- Usar os computadores para o nosso beneficio!



Introdução

Computadores

■ Problema I

- Suponha que soma (+) e subtração (-) são as únicas operações disponíveis
- Dados dois números inteiros positivos A e B , determine o **quociente** e o **resto** da divisão de A por B

Introdução

Computadores

- Qual é a dificuldade?
 - Pessoas, conseguem abstrair e entender ideias de alto nível
 - Tradicionalmente falando: **Computadores NÃO!**
 - Vamos excluir ChatGPT e afins da conversa... 😊
- Computadores são literais:
 - Precisam de uma serie de passos
 - Passos bem especificados
 - Para conseguir realizar uma tarefa qualquer

Intro

Uso da



leo
@le

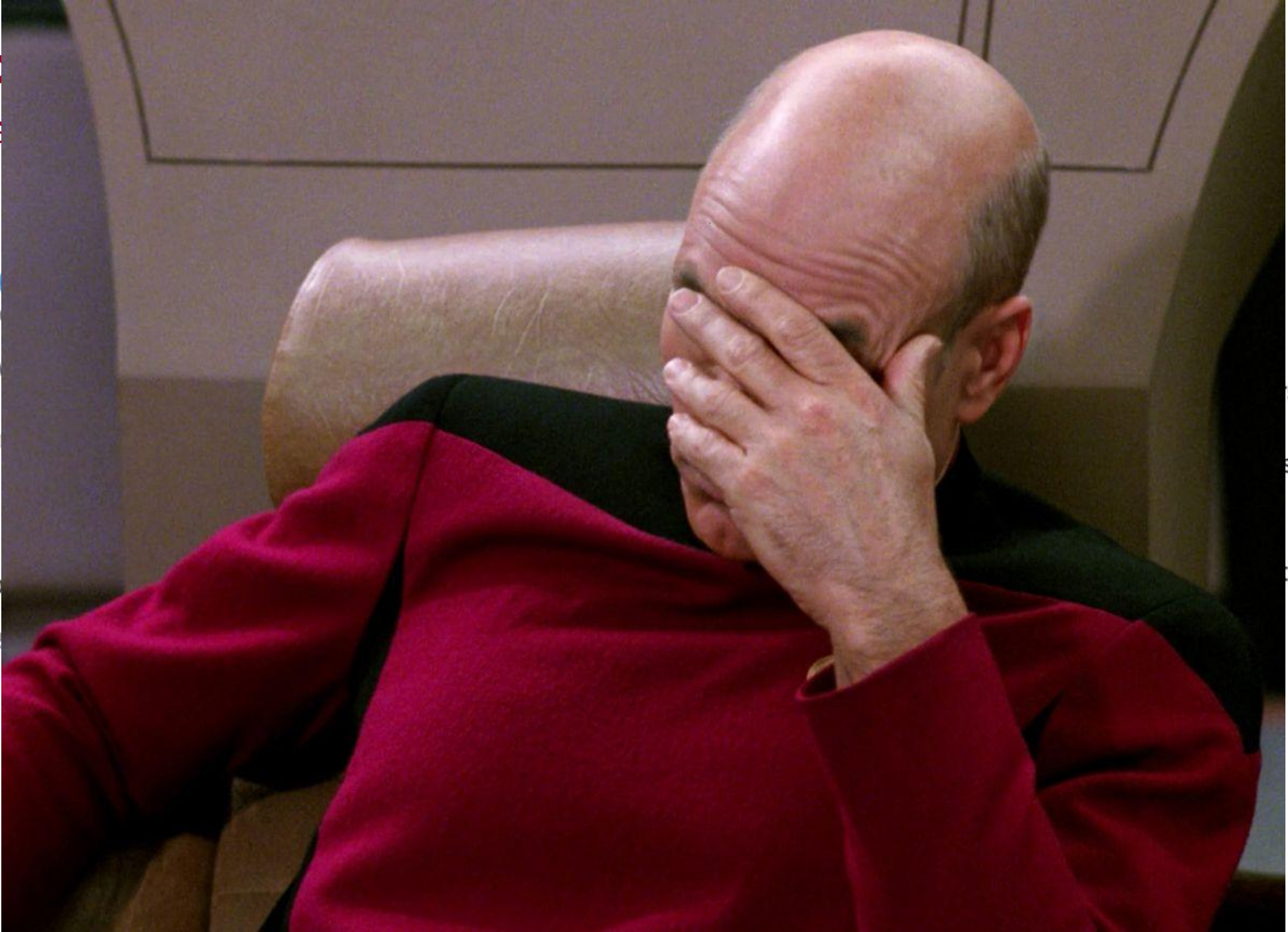
my saas w

AI is no lo

Now, you

P.S. Yes, p

4:34 am · 1



sor

s, people

that usual to

Introdução

Uso da IA, de novo...

Home → Tech → Apps & Software

Cursor AI refuses to code, tells user to learn how to do it instead



By Chris Smith 



Published Mar 14th, 2025 4:23PM EDT

Introdução

Computadores

■ Problema I

- Suponha que soma (+) e subtração (-) são as únicas operações disponíveis
- Dados dois números inteiros positivos **A** e **B**, determine o **quociente** e o **resto** da divisão de **A** por **B**

$$A=18 - 5 = 13 - 5 = 8 - 5 = 3$$

$$B=5$$

O que sobro: resto da divisão

$$1 + 1 + 1 = 3$$

Numero de subtrações: quociente da divisão

Introdução

Computadores

■ Problema I

- Suponha que soma (+) e subtração (-) são as únicas operações disponíveis
- Dados dois números inteiros positivos **A** e **B**, determine o **quociente** e o **resto** da divisão de **A** por **B**
- Para resolver o Problema I, **precisamos de um algoritmo:**

Sequência finita de **instruções** que, ao ser executada, chega a uma **solução de um problema**

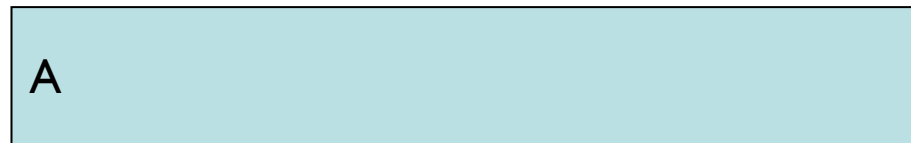
Algoritmos estruturados

- Para escrever este algoritmo, podemos usar a seguinte ideia:
 - Representar os números A e B por retângulos de larguras proporcionais aos seus valores
 - Verificar quantas vezes B cabe em A

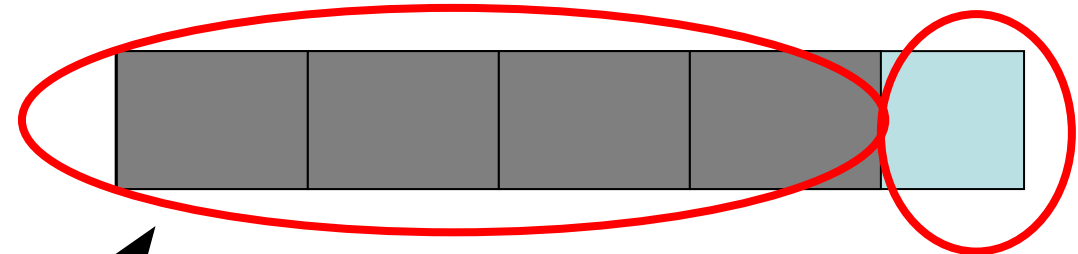
Algoritmos estruturados

Problema I

- Suponha que soma (+) e subtração (-) são as únicas operações disponíveis em C. Dados dois números inteiros positivos A e B , determine o **quociente** e o **resto** da divisão de A por B



$$A = 9, B = 2$$



Quociente de A/B : n° de vezes que B cabe em A

O que sobra em A é o resto da divisão

Algoritmos estruturados

Problema I

- Pode-se escrever este algoritmo como:

```
1. Sejam A e B os valores dados;  
2. Atribuir o valor 0 ao quociente (q);  
3. Enquanto B couber em A:  
4. {  
5.     Somar 1 ao valor de q;  
6.     Subtrair B do valor de A;  
7. }  
8. Atribuir o valor final de A ao resto (r);
```


Algoritmos estruturados

Problema I

- Pode-se escrever este algoritmo como:

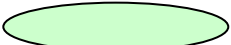
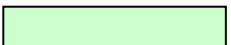
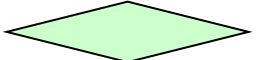



1. Sejam A e B os valores dados;
2. Atribuir o valor 0 ao quociente (q);
3. Enquanto $B \leq A$:
4. {
5. Somar 1 ao valor de q ;
6. Subtrair B do valor de A ;
7. }
8. Atribuir o valor final de A ao resto (r);

Outra forma
de representar



Fluxograma

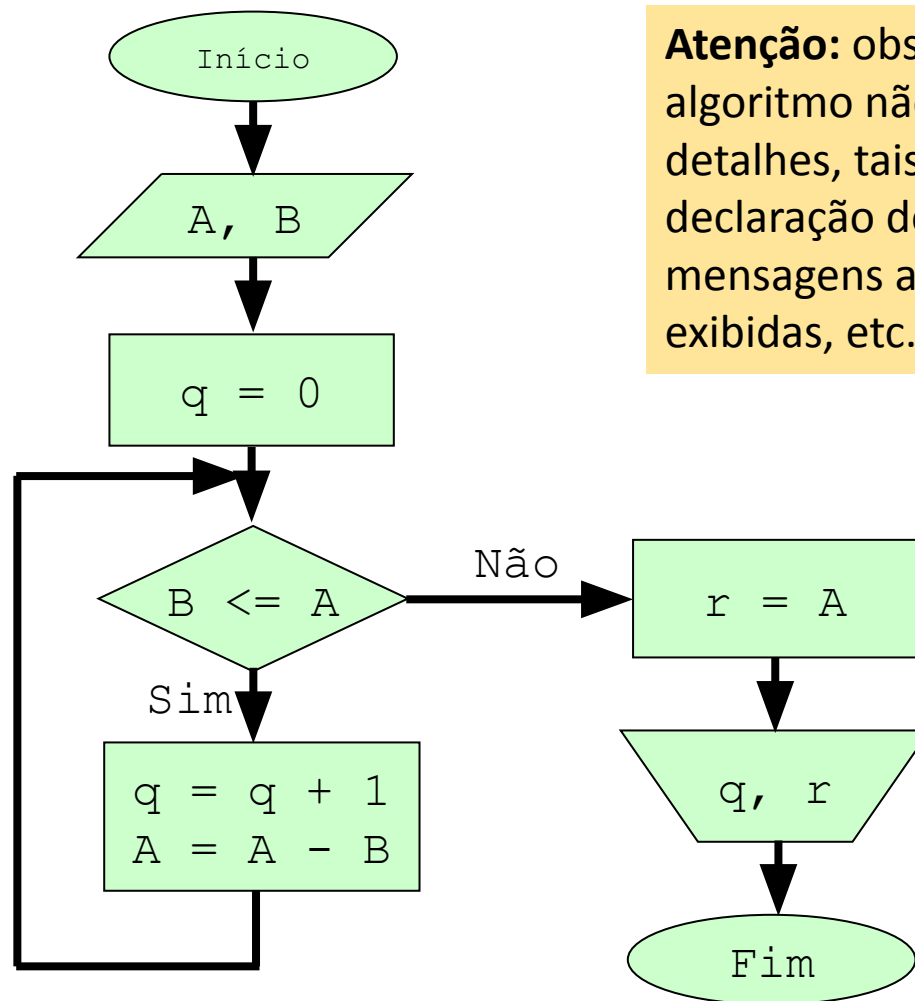
- É conveniente representar algoritmos por meio de fluxogramas (diagrama de blocos)
- Em um fluxograma, as operações possíveis são representadas por meio de figuras:

Figura	Usada para representar
	Início ou fim
	Atribuição
	Condição
	Leitura de dados
	Apresentação de resultados
	Fluxo de execução

Fluxograma

Exemplo do Problema I

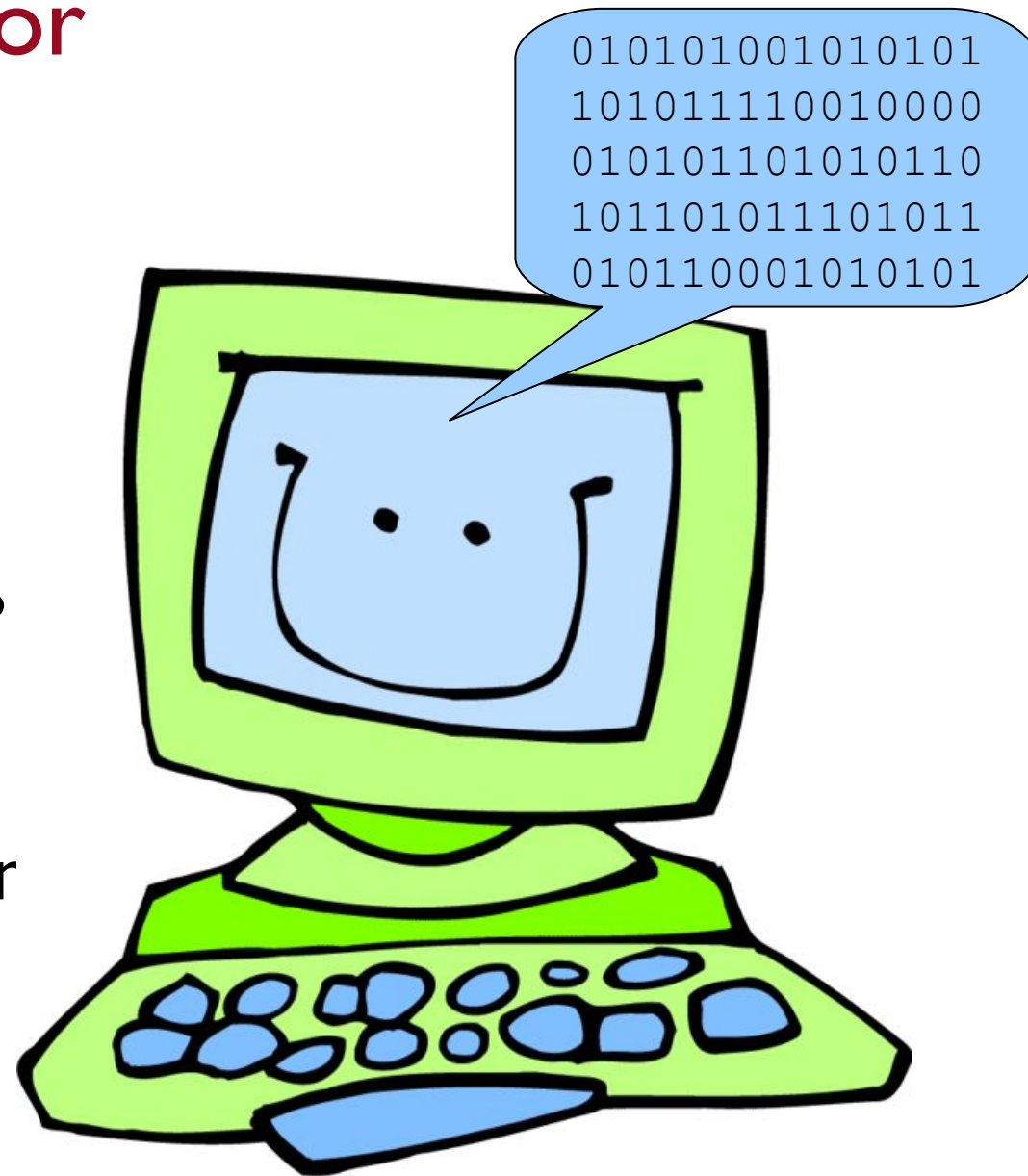
- O algoritmo para o Problema I pode ser representado pelo seguinte fluxograma
- Não estão atrelados a uma linguagem de programação específica!



Atenção: observe que um algoritmo não inclui detalhes, tais como declaração de variáveis, mensagens a serem exibidas, etc.

Conversando com o computador

- Como podemos conversar com um computador?
 - Não queremos só instâncias de problemas onde $A=10 + B=5$
 - Queremos tratar problemas onde A e B são milhões!
 - Problemas mais complexos...
- Computadores só entendem linguagem de máquina...
 - Zeros e uns!



Conversando com o computador

- Considere o seguinte problema:
 - Determinar e exibir o valor de $y = \text{seno}(1.5)$

Programa de computador

```
00010101000101010101
00101001010100101010
01011001010010101010
```

Mensagem para o computador

1. `Calcula seno(1.5)`
e armazena em `y`
2. `Imprime_na_tela(y)`
3. PAUSA

Este formato é **muito mais amigável** para humanos!

Conversando com o computador

- Considere o seguinte problema:
 - Determinar e exibir o valor de $y = \text{seno}(1.5)$

```
5 float y;  
6 y = sin(1.5);  
7 printf("seno de 1.5 eh: %f", y);  
8 printf("\n");  
9 system("PAUSE");
```

Definições

- Para resolver um problema de computação é preciso escrever um **texto**
- Este texto, como qualquer outro, obedece **regras de sintaxe**
- Estas regras são estabelecidas por uma **linguagem de programação**
- Este texto é conhecido como:

Programa



Definições

- Neste curso, será utilizada a **linguagem C**
- A **linguagem C** é subconjunto da **linguagem C++** e, por isso, geralmente, os **ambientes de programação** da linguagem C são denominados ambientes C/C++
- Um **ambiente de programação** contém:
 - **Editor de programas:** viabiliza a escrita do programa
 - **Compilador:** verifica se o texto digitado obedece à sintaxe da linguagem de programação e, caso isto ocorra, traduz o texto para uma sequência de instruções em **linguagem de máquina**

← Código binário

Ambiente de programação

- Que **ambiente de programação** iremos utilizar?
- Existem muitos ambientes de programação integrados (IDEs)
 - Code Blocks
 - VScode
 - Microsoft Visual C++
 - Borland C++ Builder
 - DEV-C++
 - Etc..



Ambiente de programação

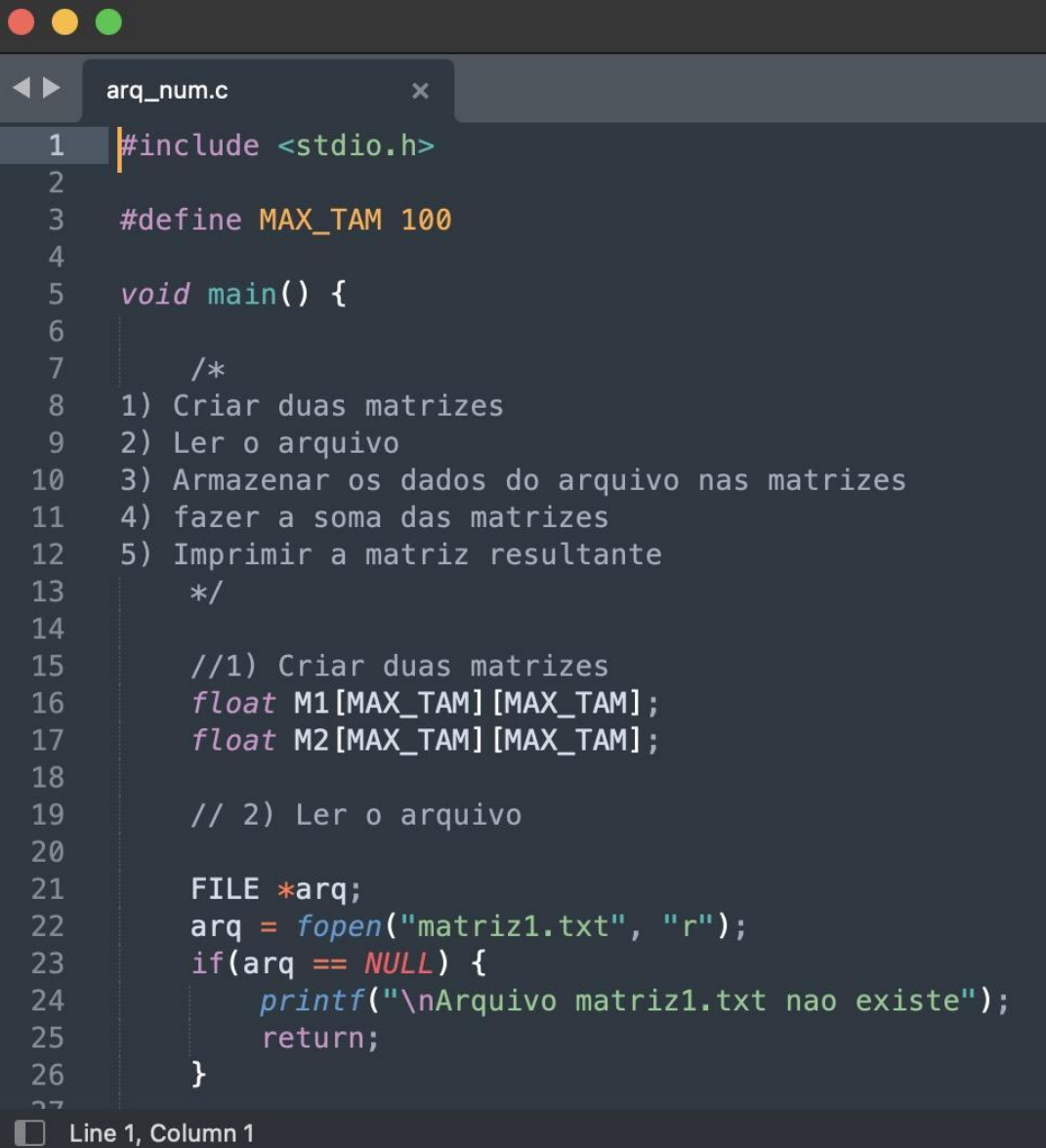
- Que **ambiente de programação** iremos utilizar?
- Nossa recomendação?
 - Nenhum! 😊
- Usar um editor de texto normal:
 - E o compilador `gcc` “na mão”



Ambiente de programação

Editores de texto gratuitos

- Editores gratuitos!
- Sublime Text:
 - <https://www.sublimetext.com>
- Notepad++:
 - <https://notepad-plus-plus.org>
- Atom:
 - <https://atom.io>
- O que precisamos é só “formatação”



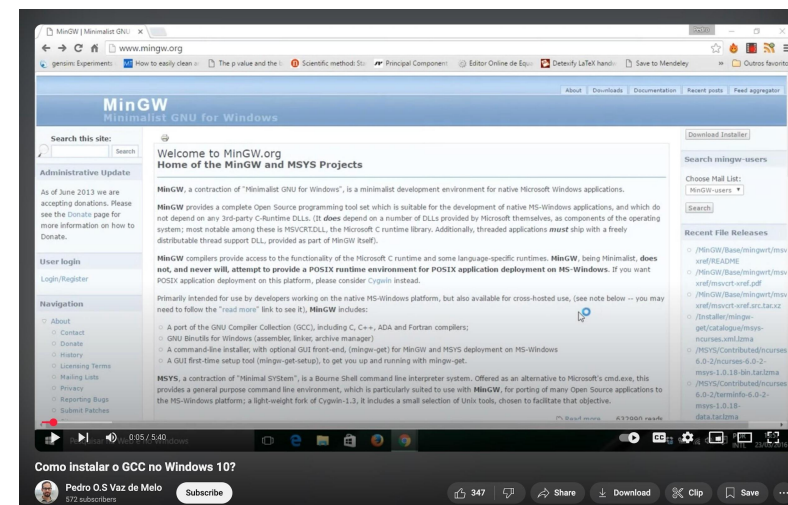
```
1  #include <stdio.h>
2
3  #define MAX_TAM 100
4
5  void main() {
6
7      /*
8      1) Criar duas matrizes
9      2) Ler o arquivo
10     3) Armazenar os dados do arquivo nas matrizes
11     4) fazer a soma das matrizes
12     5) Imprimir a matriz resultante
13     */
14
15     //1) Criar duas matrizes
16     float M1[MAX_TAM][MAX_TAM];
17     float M2[MAX_TAM][MAX_TAM];
18
19     // 2) Ler o arquivo
20
21     FILE *arq;
22     arq = fopen("matriz1.txt", "r");
23     if(arq == NULL) {
24         printf("\nArquivo matriz1.txt nao existe");
25         return;
26     }
27 }
```

Line 1, Column 1

Ambiente de programação

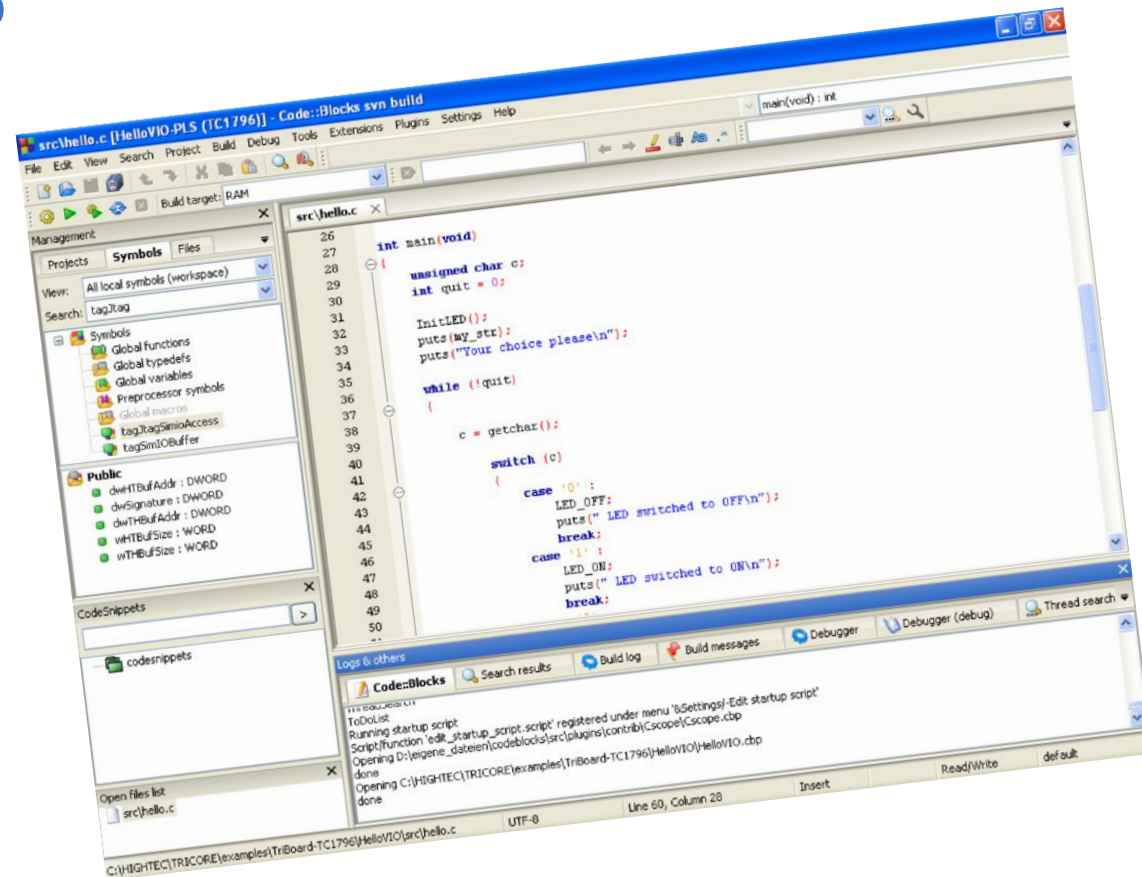
Instalação GCC (Windows)

- O GCC é o compilador de C que vamos usar na aula
- Linux já tem o compilador instalado por padrão:
 - Se não é só rodar o comando
`sudo apt get install gcc`
- Em Windows o processo é diferente:
 - O Prof Pedro Olmo fez um vídeo tutorial explicando o passo a passo
 - <https://www.youtube.com/watch?v=FzPBZjkoEmA>



Ambiente de programação

- Que **ambiente de programação** iremos utilizar?
- Se quiserem usar uns IDE mais avançados:
 - Vscode
 - Codeblocks
 - DEV-C++
- Pela sua conta e risco!
- Eles integram edição e compilação
- Não é necessário nessa aula...



Definições

- Porque o compilador traduz o programa escrito na **linguagem de programação** para a **linguagem de maquina**?

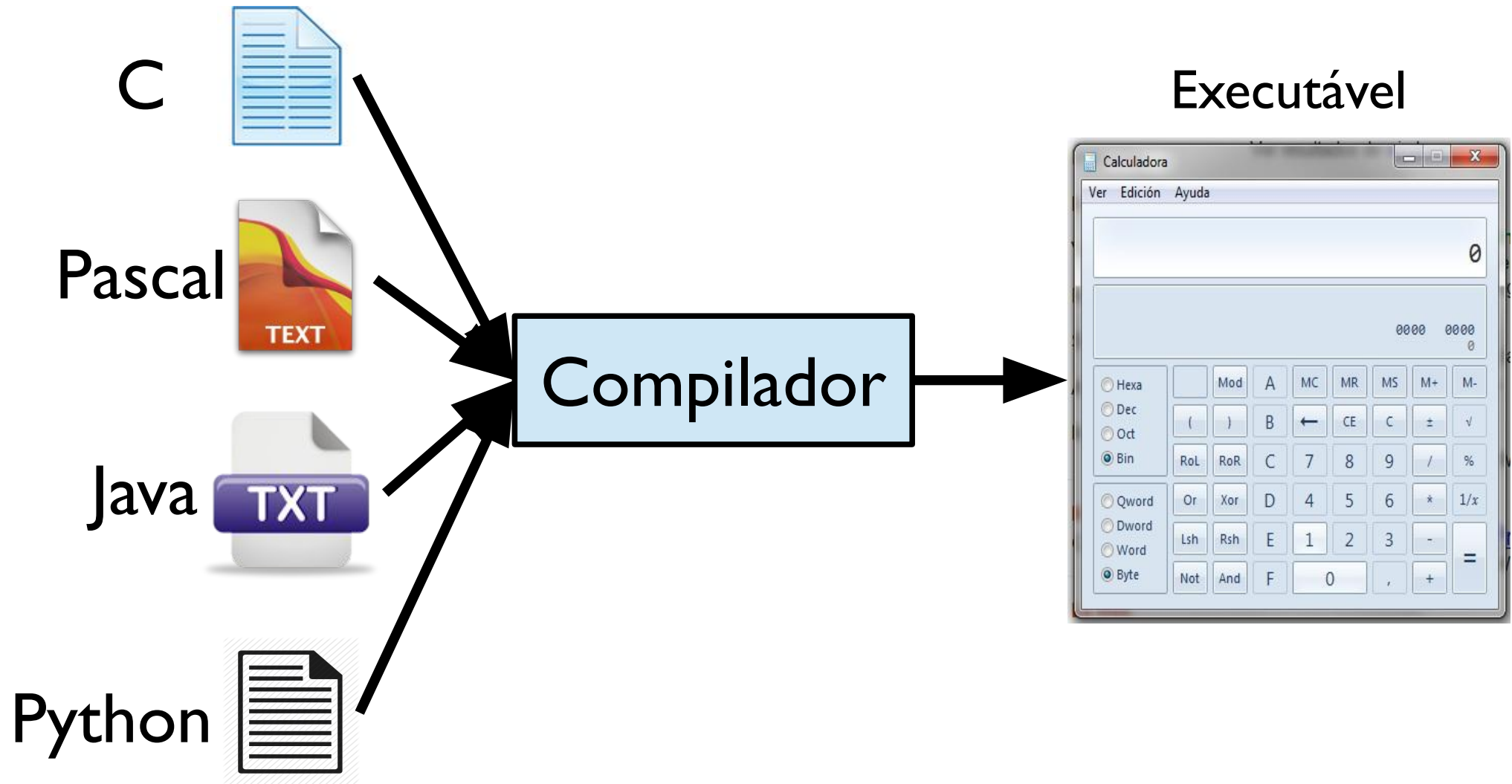
```
1 #include <stdio.h>
2 #include <math.h>
3
4 int main(int argc, char* argv[]) {
5     float y;
6     y = sin(1.5);
7     printf("seno de 1.5 eh: %f", y);
8     printf("\n");
9     system("PAUSE");
10    return 0;
11 }
```

Compilador

```
0101010110100010011
1000101010111101111
1010100101100110011
0011001111100011100
0101010110100010011
1000101010111101111
1010100101100110011
0011001111100011100
```

- Os computadores atuais só conseguem executar instruções que estejam escritas na forma de códigos binários
- Um programa em **linguagem de máquina** é chamado de **programa executável**

Definições



Compilando o nosso primeiro programa

Exemplo `hello`

Exemplo VSCODE

Erros de sintaxe

- Atenção!

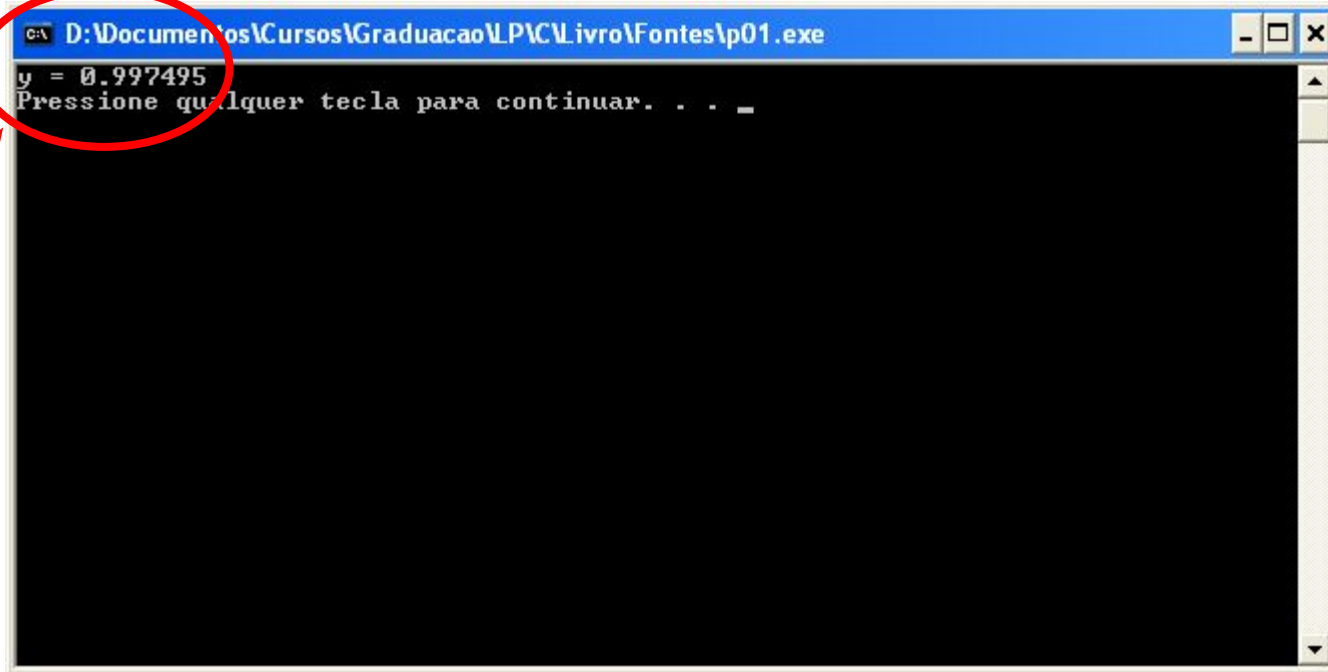
- O **programa executável** só será gerado se o texto do programa não contiver **erros de sintaxe**
 - Exemplo: considere uma **string**
 - Ah?! O que é isso?! Uma sequência de caracteres delimitada por aspas
 - Se isso é uma **string** e se tivéssemos escrito:
- O compilador iria apontar um erro de sintaxe nesta linha do programa e exibir uma mensagem tal como:

```
printf("y = %f", y);
```

```
undetermined string or character constant
```

Erros de sintaxe

- Se o nome do programa é `p1.c`, então após a compilação será produzido o programa executável `p1.exe` (ou `a.exe`)
- Executando-se o programa `p1.exe`, o resultado será:



Problema Resolvido!

Erros de lógica

- Atenção!

- Não basta obter o programa executável!! Será que ele está correto?
- Se ao invés de: `y = sin(1.5);`
- Tivéssemos escrito: `y = sin(2.5);`
- O compilador também produziria o programa `p1.exe`, que executado, iria produzir:



```
C:\ D:\Documentos\Cursos\Graduacao\LPIC\Livro\Fontes\p01.exe
y = 0.598472
Pressione qualquer tecla para continuar. . . _
```

Erros de lógica

- Embora um resultado tenha sido obtido, **ele não é correto**
- Se um programa executável não produz os resultados corretos, é porque ele contém **erros de lógica ou bugs**
- O processo de identificação e correção de erros de lógica é denominado **depuração (debug)**
- O nome de um texto escrito em uma linguagem de programação é chamado de **programa-fonte**
 - Exemplo: o programa `p1.c` é um **programa-fonte**

Arquivos de cabeçalho

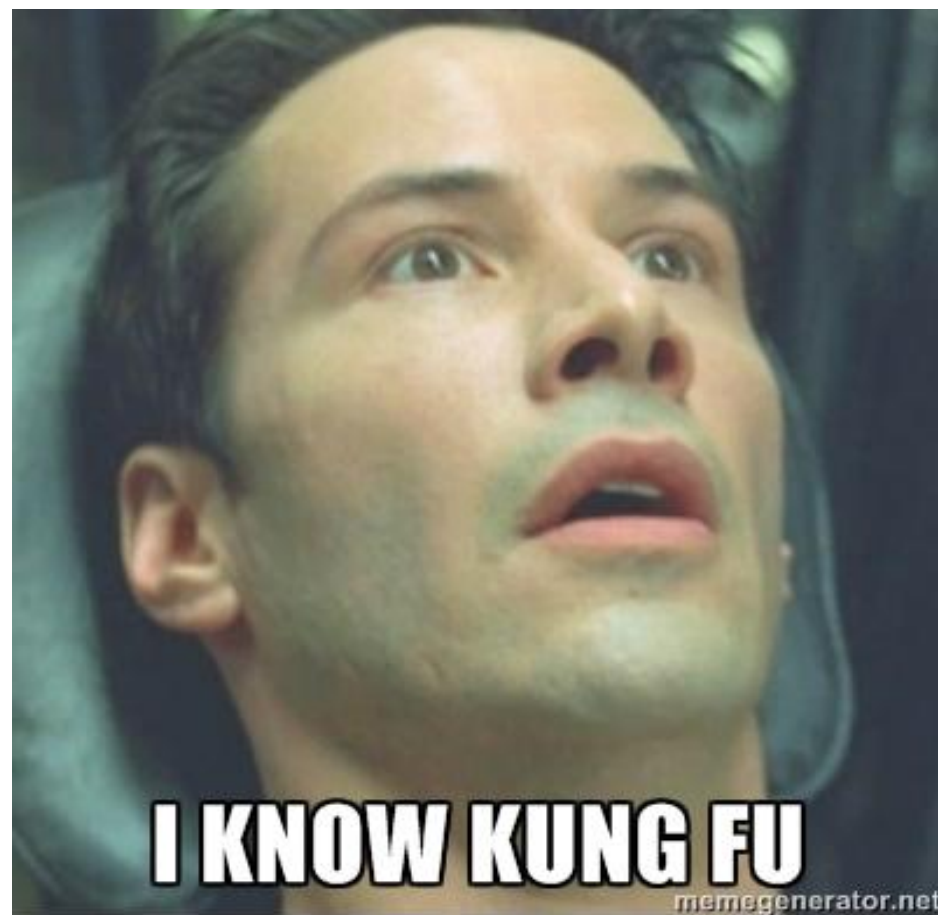
- Note que o programa-fonte `p1.c` começa com as linhas:

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

- Todo programa-fonte em linguagem C começa com linhas deste tipo.
- O que elas indicam?
 - Dizem ao compilador que o programa-fonte vai utilizar arquivos de cabeçalho (extensão `.h`, de header)
 - E daí? O que são estes arquivos de cabeçalho?
 - Eles **contêm informações** que o compilador precisa para construir o programa executável

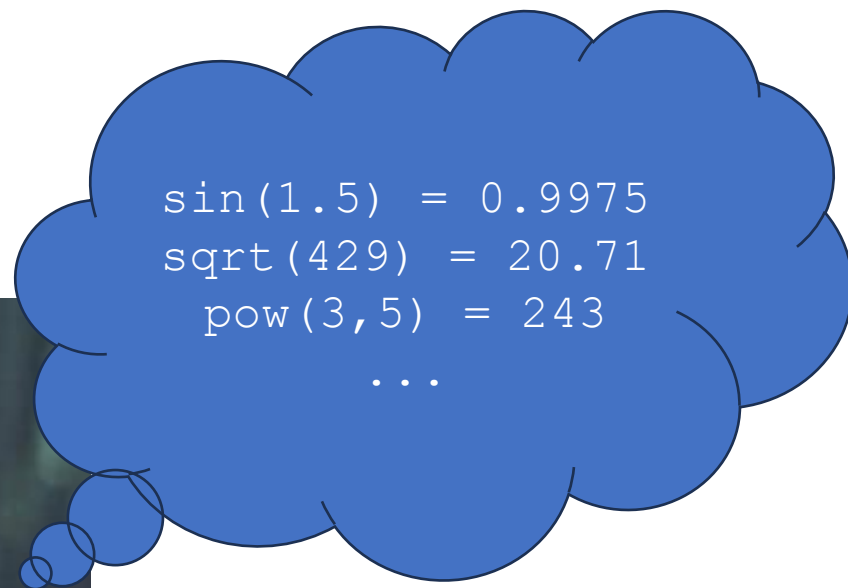
Arquivos de cabeçalho

```
#include <kungfu.h>
```



Arquivos de cabeçalho

```
#include <math.h>
```



I KNOW MATH!!!

Arquivos de cabeçalho

- Como assim?
- Observe que o programa `p1.c` inclui algumas **funções**, tais como:
 - **sin** – função matemática seno
 - **printf** – função para exibir resultados
- Por serem muito utilizadas, a linguagem C mantém funções como estas em **bibliotecas**
- Atenção!
 - O conteúdo de um arquivo de cabeçalho também é um texto

Arquivos de cabeçalho

- Ao encontrar uma instrução `#include` em um programa-fonte, o compilador traduz este texto da mesma forma que o faria se o texto tivesse sido digitado no programa-fonte

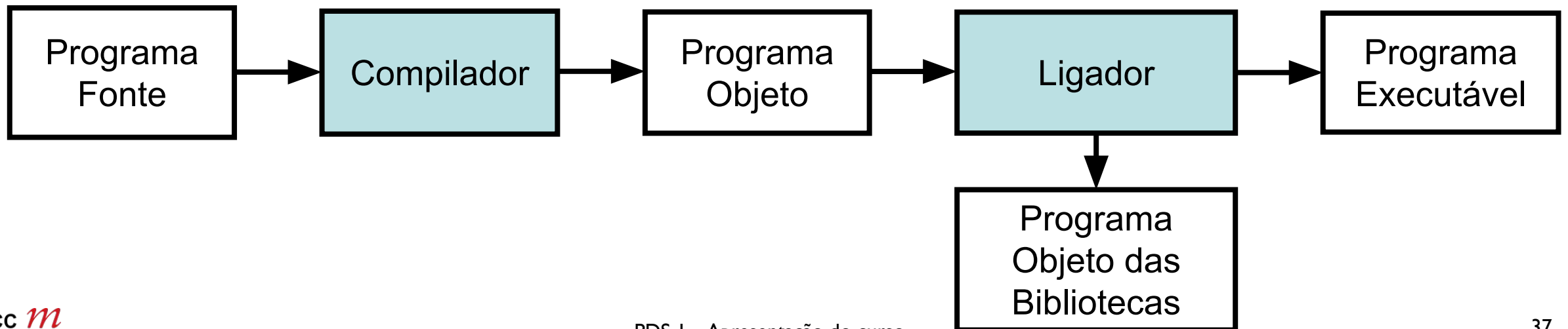
- Portanto, as linhas:

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

- Indicam ao compilador que o programa `p1.c` utilizará as instruções das bibliotecas `stdio` e `stdlib`

Processo de compilação

- O processo de compilação, na verdade, se dá em duas etapas:
 - **Fase de tradução:** programa-fonte é transformado em um programa-objeto
 - **Fase de ligação:** junta o programa-objeto às instruções necessárias das bibliotecas para produzir o **programa executável**



Função main

- A próxima linha do programa é:

```
int main(int argc, char *argv[])
```

- Esta linha corresponde ao cabeçalho da função **main** (a função principal, daí o nome **main**)
- O texto de um programa em Linguagem C pode conter muitas outras funções e **SEMPRE deverá conter a função main**

int	main	(int argc, char *argv[])
-----	------	--------------------------

Tipo de valor
produzido
pela função

Nome da
função

Lista de
parâmetros
da função

Função `main`

- A Linguagem C é **case sensitive**. Isto é, considera as letras maiúsculas e minúsculas diferentes
- Atenção!
 - O nome da função principal deve ser escrito com letras minúsculas: `main`
 - `Main` ou `MAIN`, por exemplo, **provocam erros de sintaxe**
- Da mesma forma, as palavras `int` e `char`, devem ser escritas com letras minúsculas

Perguntas?

- E-mail:
 - hector@dcc.ufmg.br
- Material da disciplina:
 - <https://pedroolmo.github.io/teaching/pdsI.html>



Héctor Azpúrua
h3ct0r