

**VIANNA JUNIOR**  
**INSTITUTO**



# Algoritmos

## Introdução e Conceitos Básicos

Professor: Camillo Falcão

# Organização Básica de um Computador



*Processador*



*Memória principal*

*Canal de comunicação*

*Memória secundária*



*Dispositivos de Entrada*



*Dispositivos de Saída*



- Principal componente de um computador.
- Executa sequências de operações muito simples e precisas, sempre uma por vez.
- Muito rápido na execução de operações (i7 é capaz de executar cerca de 112.000.000.000 operações matemáticas por segundo)



# Memória Principal (RAM – Random Access Memory)

- É a unidade encarregada de armazenar os programas e dados que estão sendo processados.
- É considerada um meio temporário de armazenamento de dados, pois estes permanecem ali somente durante o tempo em que estiverem sendo utilizados pelo processador.
- É na memória principal que o processador armazena os resultados de cada uma das operações que ele realiza.



# Memória Secundária

- Pode ser composta por vários dispositivos capazes de armazenar grandes quantidades de dados e programas.
- É um tipo de memória não volátil, teoricamente permanente e mais lenta.
- Um programa armazenado em memória secundária precisa ser carregado na memória principal antes de ser executado.





# Dispositivos de Entrada

- São os recursos ou componentes que permitem que o usuário forneça dados para o computador.
- Nesta disciplina, o dispositivo padrão para entrada de dados nos programas é o teclado.



# Dispositivos de Saída

- São os recursos ou componentes que permitem que o computador forneça dados para o usuário.
- Nesta disciplina, o dispositivo padrão para saída de dados nos programas é o monitor.





# Canal de Comunicação

- A comunicação entre os dispositivos é realizada através de diferentes tipos de cabos que, em geral, transmitem um sinal contínuo.
- Dados são transmitidos utilizando apenas dois estados possíveis: ausência ou presença do sinal a cada instante.



Bit (“Binary DigiT” – dígito binário)

- Menor unidade de Informação, que armazena somente um dos valores “0” ou “1”;

Byte (“BinarY Term” – termo binário)

- Conjunto de 8 bits, com o qual pode-se representar os números, as letras, os sinais de pontuação, etc...

Palavra (Word)

- É a quantidade de bits que a CPU processa por vez.
- Nos computadores atuais, são comuns palavras de 32 ou 64 bits.

Unidades	Usual	Informática
Kilo (K)	$10^3$	$2^{10}$ bytes
Mega (M)	$10^6$	$2^{20}$ bytes
Giga (G)	$10^9$	$2^{30}$ bytes
Tera (T)	$10^{12}$	$2^{40}$ bytes

Exemplo: Qual a quantidade exata de bits que uma PEN DRIVE de 8 GB possui?

$$8 \text{ GB} = 8 * 2^{30} * 8 = 68719476736 \text{ bits}$$

Representação de uma memória de 1 KB:

Endereço	Byte							
0								
1								
2	0	1	0	0	0	0	0	1
...								
1023								

- No byte com endereço 2 está armazenado o código binário que representa o caractere “A”.
- O processador acessa o conteúdo de um byte a partir do endereço desse byte.

- Os computadores só trabalham com bytes e palavras.
- Toda informação armazenada e processada é um conjunto de bits e bytes:
  - Letras, dígitos e símbolos;
  - Cores, ícones, figuras e fotos;
  - Textos, músicas e vídeos...



# Bits e Bytes

Dec	Binário		Dec	Binário		Dec	Binário	
32	00100000		64	01000000	@	96	01100000	`
33	00100001	!	65	01000001	A	97	01100001	a
34	00100010	"	66	01000010	B	98	01100010	b
35	00100011	#	67	01000011	C	99	01100011	c
36	00100100	\$	68	01000100	D	100	01100100	d
37	00100101	%	69	01000101	E	101	01100101	e
38	00100110	&	70	01000110	F	102	01100110	f
39	00100111	'	71	01000111	G	103	01100111	g
40	00101000	(	72	01001000	H	104	01101000	h
41	00101001	)	73	01001001	I	105	01101001	i
42	00101010	*	74	01001010	J	106	01101010	j
43	00101011	+	75	01001011	K	107	01101011	k
44	00101100	,	76	01001100	L	108	01101100	l
45	00101101	-	77	01001101	M	109	01101101	m
46	00101110	.	78	01001110	N	110	01101110	n
47	00101111	/	79	01001111	O	111	01101111	o
48	00110000	0	80	01010000	P	112	01110000	p
49	00110001	1	81	01010001	Q	113	01110001	q
50	00110010	2	82	01010010	R	114	01110010	r
51	00110011	3	83	01010011	S	115	01110011	s
52	00110100	4	84	01010100	T	116	01110100	t
53	00110101	5	85	01010101	U	117	01110101	u
54	00110110	6	86	01010110	V	118	01110110	v
55	00110111	7	87	01010111	W	119	01110111	w
56	00111000	8	88	01011000	X	120	01111000	x
57	00111001	9	89	01011001	Y	121	01111001	y
58	00111010	:	90	01011010	Z	122	01111010	z
59	00111011	;	91	01011011	[	123	01111011	{
60	00111100	<	92	01011100	\	124	01111100	
61	00111101	=	93	01011101	]	125	01111101	}
62	00111110	>	94	01011110	^	126	01111110	~
63	00111111	?	95	01011111	_			

- Tabela ASCII: padrão para letras, dígitos e símbolos.

# Bits, bytes e programas

- O computador é capaz de executar diferentes programas, desenvolvidos com finalidades distintas: processador de texto, planilha eletrônica, calculadora, navegador, etc.
- Cada programa só pode ser executado através de um arquivo executável.



# Bits, bytes e programas

- Um arquivo executável é composto por milhares de instruções simples definidas através de sequências de 0's e 1's.
- A linguagem que define a estrutura de um arquivo executável é denominada *Linguagem de Máquina*.

ADD	0001	DR	SR1	0	00	SR2
ADD	0001	DR	SR1	1	imm5	
AND	0101	DR	SR1	0	00	SR2
AND	0101	DR	SR1	1	imm5	
NOT	1001	DR	SR	111111		
JMP	1100	0	00	BaseR	000000	

# Linguagem de máquina

- A *Linguagem de Máquina* é a linguagem que o computador é capaz de entender e executar.
- Por se tratar de uma sequência muito grande de 0's e 1's, programar em Linguagem de Máquina é uma tarefa extremamente difícil.

ADD	0001	DR	SR1	0	00	SR2
ADD	0001	DR	SR1	1	imm5	
AND	0101	DR	SR1	0	00	SR2
AND	0101	DR	SR1	1	imm5	
NOT	1001	DR	SR	111111		
JMP	1100	0	00	BaseR	000000	

- Um grande avanço ocorreu na computação quando começaram a surgir programas que traduziam instruções para Linguagem de Máquina.
- Instruções em Linguagens de Programação (ou Linguagens de Alto Nível) são escritas de forma muito mais clara e legível para o programador.

O Programador escreve um programa em uma **Linguagem de Programação**.

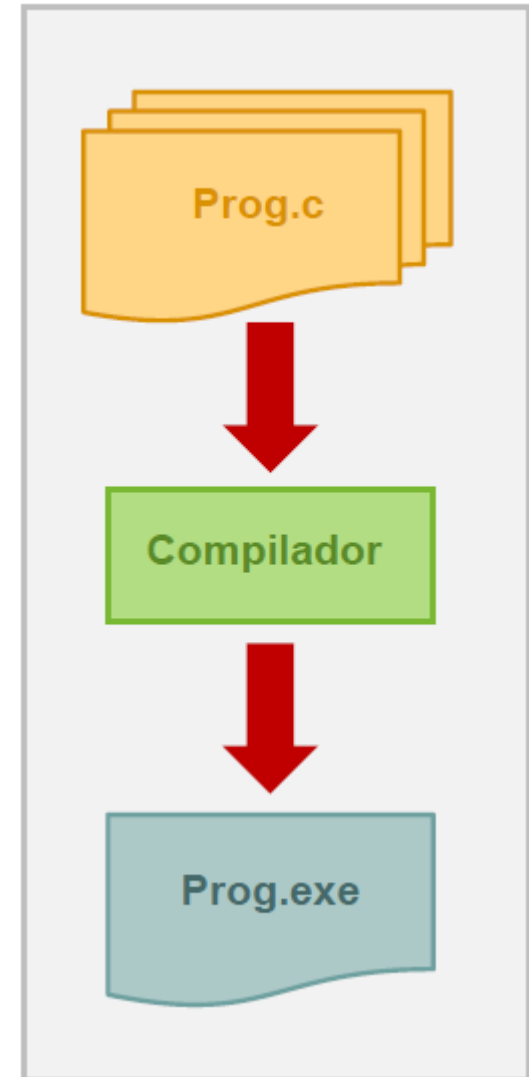
Um programa específico é utilizado para traduzir as instruções definidas em Linguagem de Programação.

O resultado é um programa em **Linguagem de Máquina**.



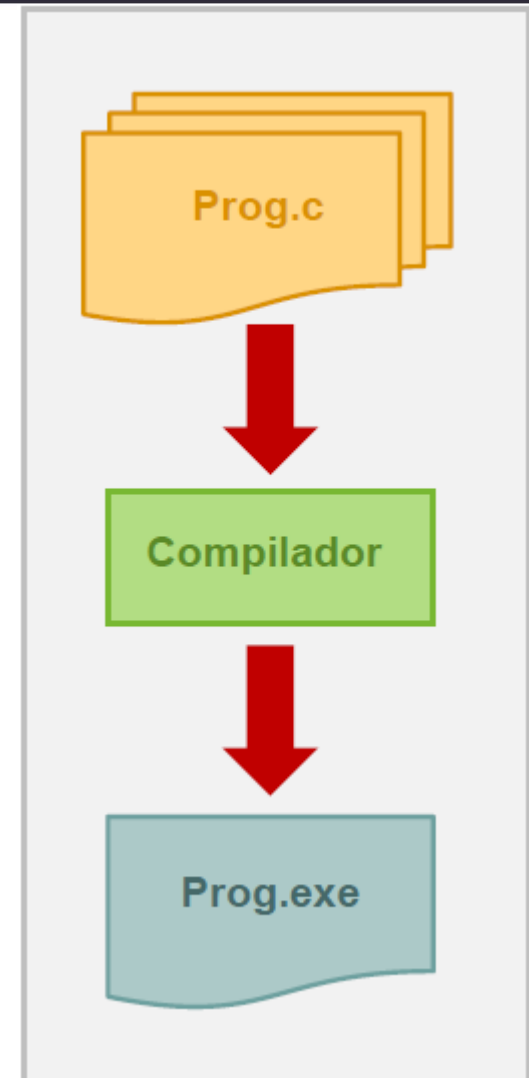
Nesta disciplina:

- Linguagem de Programação: linguagem C
- Código fonte: arquivos com extensão “.c”
- Tradutor: compilador
- Traduzir: compilar



Para que o processo de tradução seja possível, é necessário que o compilador consiga identificar cada instrução no código fonte.

Por este motivo, o programador precisa observar uma série de regras ao utilizar uma linguagem de programação.



## SINTAXE:

- Conjunto de regras que definem como uma linguagem de programação pode ser utilizada.
- O programador precisa seguir estas regras ao escrever as instruções dos programas para que o compilador possa “entendê-las” e traduzí-las.



## SEMÂNTICA:

- Cada instrução tem uma finalidade bem específica.
- A combinação de instruções tem um significado lógico, isto é, uma sequência de instruções colabora de forma parcial para que os objetivos do programa sejam alcançados durante sua execução.

## SINTAXE:

- Se o programador comete um ***erro de sintaxe***, o compilador interrompe o processo de tradução e indica a linha do arquivo onde o erro provavelmente ocorreu (o arquivo executável não é gerado).
- Exemplos de erros de sintaxe:
  - Palavras com erro de grafia;
  - Parênteses ou aspas que não fecham;
  - Ausência de vírgulas, pontos ou ponto e vírgula;
  - Entre outros.

## SEMÂNTICA:

- Se o programa gerado não cumpre ou cumpre parcialmente seus objetivos, seu código fonte provavelmente contém ***erros de lógica***.
- É comum um programa funcionar corretamente na maioria das vezes e apresentar um erro de lógica só em condições muito específicas.
- Erros de lógica não são apontados pelo compilador; encontrá-los é tarefa do programador.
- Estes erros costumam ser difíceis de se encontrar...

Qual o erro de lógica na sequência de instruções do programa abaixo?

1. Peça ao usuário do programa que digite o número de pontos do time 1 (chamaremos este valor de pt1)
2. Peça ao usuário que digite o número de pontos do time 2 (chamaremos este valor de pt2)
3. Se  $pt1 > pt2$ :
  - 3.1. Escreva “Time 1 venceu”
4. Senão:
  - 4.1. Escreva “Time 2 venceu”

No desenvolvimento de um programa, estes dois aspectos estão presentes todo o tempo:

## SINTAXE:

é necessário conhecer a linguagem de programação e suas regras para poder construir o programa.

## SEMÂNTICA:

é necessário conhecer a finalidade de cada instrução da linguagem e, principalmente, é necessário saber combinar estas finalidades isoladas para se alcançar o objetivo do programa.



# Lógica de programação

- No desenvolvimento de um programa, o programador utiliza um modo de raciocínio pouco comum em outras áreas do conhecimento:

*encontrar uma sequência de instruções que resolvam um determinado problema.*

- A Lógica de Programação pode ser entendida como o conjunto de raciocínios utilizados nesta tarefa.

## ***Algoritmo***

Sequência de instruções que resolve determinado problema.

## ***Programa***

Algoritmo escrito em uma linguagem de programação específica, isto é, um algoritmo que pode ser executado em um computador.

## ***Lógica de Programação***

Conjunto de raciocínios utilizados para criar um algoritmo.



- De forma mais detalhada, um algoritmo:
  - É uma sequência de instruções bem definidas;
  - Pode receber ou gerar informações (dados de entrada e saída);
  - Tem um início e é finito (sempre termina);
  - Cumpre um propósito específico.

- Na maior parte das vezes, elaborar o algoritmo para resolver um problema é o maior desafio na programação.
- Embora algoritmos já façam parte do dia a dia das pessoas, elaborar algoritmos de forma sistemática é uma atividade raramente exercitada.

## Treino de corrida para iniciantes

1. Caminhe por 5 minutos em ritmo lento;
2. Repita 5 vezes a seguinte sequência:
  - 2.1. Corra por 30 segundos em um ritmo em que você respire com dificuldade;
  - 2.2. Caminhe por 60 segundos.
3. No final, caminhe por 5 minutos em ritmo lento.

## Receita de brigadeiro

1. Coloque em uma panela o conteúdo de uma lata de leite condensado, três colheres de sopa de chocolate em pó e uma colher de sopa de manteiga;
2. Misture bem;
3. Leve ao fogo baixo;
4. Até que o brigadeiro comece a desprender do fundo da panela:
  - 4.1. Mexa o brigadeiro para que não grude na panela;
5. Retire do fogo;
6. Coloque em um prato untado com manteiga;
7. Enquanto houver brigadeiro no prato:
  - 7.1. Retire uma parte do brigadeiro com uma colher pequena;
  - 7.2. Faça uma bolinha;
  - 7.3. Passe a bolinha sobre o chocolate granulado;
  - 7.4. Coloque em uma forminha.

Observe que (como vimos na aula anterior):

- A ordem das instruções é significativa.
- Dificilmente existe um único algoritmo para resolver um problema.

- Vamos tentar?

*Elabore um algoritmo para trocar uma lâmpada em um quarto vazio, assumindo que estão disponíveis uma escada e uma lâmpada nova.*

## Como trocar uma lâmpada comum em um quarto vazio

1. Se o interruptor está ligado:
  - 1.1. Desligue o interruptor;
2. Pegue a escada;
3. Leve a escada até o local;
4. Posicione a escada;
5. Busque a lâmpada nova;
6. Suba um degrau;
7. Enquanto não alcançar a lâmpada queimada e houver degrau acima:
  - 7.1. Suba um degrau;
8. Retire a lâmpada queimada;
9. Coloque a lâmpada nova;
10. Enquanto não chegar no chão:
  - 10.1. Desça um degrau;
11. Acione o interruptor;
12. Jogue a lâmpada queimada no lixo apropriado;
13. Guarde a escada.



- Outro exercício:

*Um senhor está em uma das margens de um rio com uma raposa, um saco de milho e uma dúzia de galinhas. Ele precisa atravessar e dispõe de uma canoa que suporta apenas seu peso juntamente com uma de suas cargas. Ele não pode deixar a raposa sozinha com as galinhas em uma das margens. Também não pode deixar as galinhas sozinhas com o milho. Que instruções você daria para o senhor atravessar o rio?*

## Travessia do rio

1. Atravesse com as galinhas;  
(se levasse a raposa, as galinhas ficariam com o milho;  
se levasse o milho, a raposa ficaria com as galinhas)
2. Deixe as galinhas e retorne sozinho;
3. Atravesse com a raposa;
4. Deixe a raposa e retorne com as galinhas;  
(as galinhas não poderiam ficar com a raposa)
5. Deixe as galinhas e atravesse com o milho;  
(não adiantaria voltar com as galinhas)
6. Deixe o milho e retorne sozinho;
7. Atravesse com as galinhas.

Comentários  
não fazem parte do  
algoritmo, mas ajudam  
outros programadores a  
entendê-lo.

- Ao longo do curso, vamos resolver:
  - Problemas matemáticos, como, por exemplo, média aritmética de uma sequência numérica, raízes de uma equação de segundo grau, máximo divisor comum entre dois números, operações em matrizes, etc.

- Ao longo do curso, vamos resolver:
  - Questões genéricas de leitura, armazenamento e processamento de dados, como, por exemplo, atualizar o saldo de uma conta, encontrar os dados de uma pessoa pelo seu nome, verificar quantas pessoas são maiores de idade em um evento, etc.
  - Entre outros.

# Algoritmos

## Formas de representação

- Dentre as formas de representação de algoritmos mais conhecidas, sobressaltam:
  - A Descrição Narrativa
  - O Fluxograma Convencional
  - O Diagrama de Chapin
  - A Pseudolinguagem

# Descrição narrativa

- Nesta forma de representação, os algoritmos são expressos diretamente em **linguagem natural** (como fizemos até agora).

## Troca de um pneu furado

1. Afrouxe ligeiramente as porcas;
2. Suspenda o carro;
3. Retire as porcas e o pneu;
4. Coloque o pneu reserve;
5. Aperte as porcas;
6. Abaixar o carro;
7. Dê o aperto final nas porcas.



## Cálculo da média de um aluno

1. Obter as notas da primeira e da segunda prova;
  2. Somar as duas notas e dividir o resultado por 2;
  3. Se a média for maior ou igual a 6,
    - 3.1. Escreve “Aprovado”;
  4. Senão,
    - 4.1. Escreve “Reprovado”.
- Esta representação precisa ser usada com cuidado porque o uso de linguagem natural pode dar oportunidade a más interpretações, ambigüidades e imprecisões.

- Representação gráfica de algoritmos onde formas geométricas diferentes indicam instruções distintas.
- Tal propriedade facilita o entendimento das idéias contidas nos algoritmos.

Principais formas geométricas usadas em fluxogramas.



= Início e final do fluxograma .



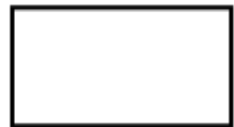
= Operação de entrada de dados



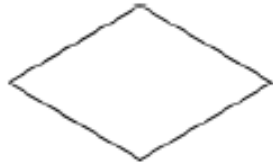
= Operação de saída de dados em impressora



= Operação de saída de dados em vídeo



= Operações de atribuição



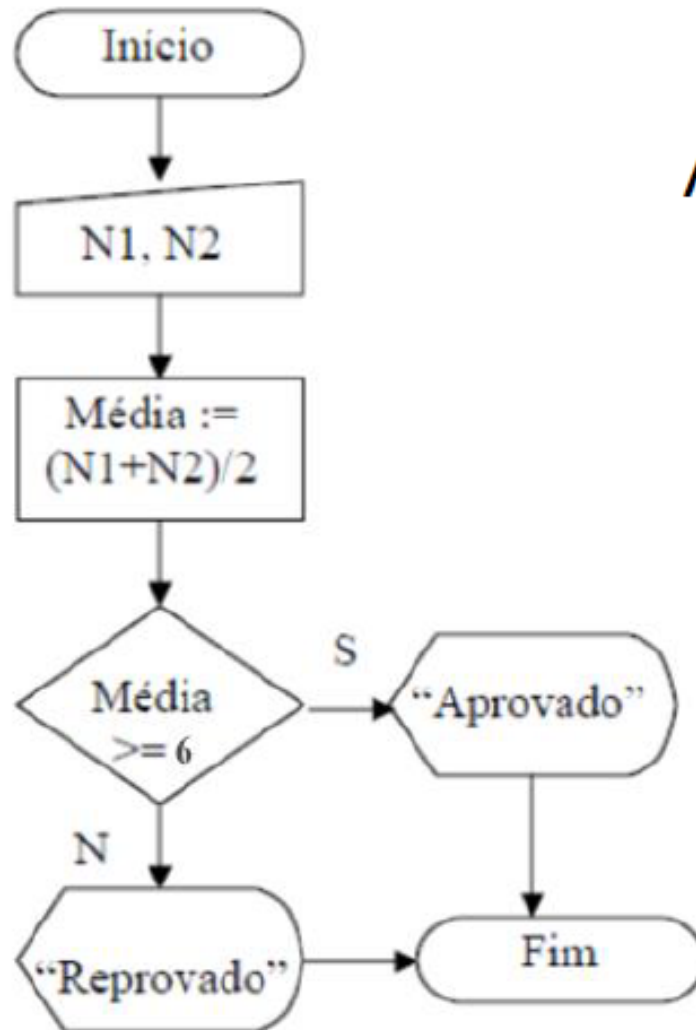
= Decisão



= Seta do fluxo de dados



= Conector utilizado quando é preciso particionar o diagrama, colocando uma letra ou número no símbolo para indentificar os pares da conexão



A figura ao lado mostra a representação do algoritmo para cálculo da média de um aluno sob a forma de um fluxograma.

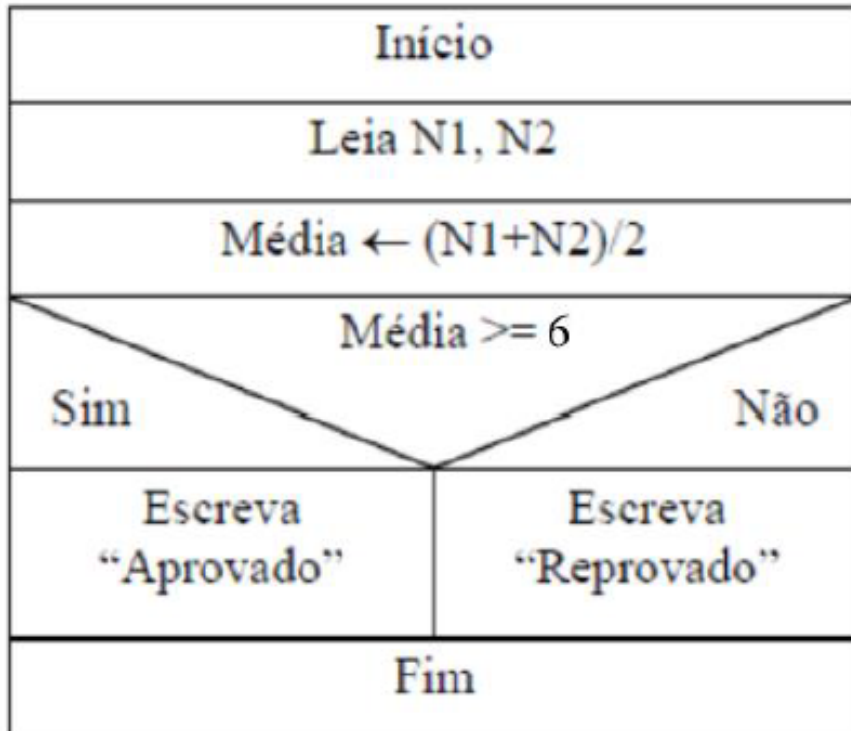
- Desenhar o fluxograma é uma tarefa difícil dependendo da complexidade do algoritmo.
- Modificar ou corrigir o algoritmo, depois de desenhado o fluxograma, pode ser complicado.



# Diagrama de Chapin

- O diagrama foi criado por Ned Chapin.
- Surgiu para substituir o fluxograma tradicional possibilitando uma visão hierárquica e estruturada da lógica do programa.

# Diagrama de Chapin



Na figura ao lado encontra-se um diagrama de Chapin representando o algoritmo para cálculo da média de um aluno.

# Diagrama de Chapin

- Representar o diagrama é difícil, especialmente quando há muitos comandos aninhados (isto é, quando um item do algoritmo tem um subitem, que tem um subitem...).
- Assim como ocorre no fluxograma, modificar ou corrigir o algoritmo pode ser complicado.

# Pseudolinguagem

- Esta forma de representação de algoritmos, também conhecida como **pseudocódigo**, **português estruturado** ou **portugol**, é bastante rica em detalhes e assemelha-se bastante à forma em que os programas são escritos.
- A tradução do pseudocódigo de um algoritmo para uma linguagem de programação é praticamente direta.

# Pseudolinguagem

O pseudocódigo que representa o algoritmo de cálculo da média de um aluno:

```
principal
{
    real n1, n2, media;
    leia(n1, n2);
    media  $\leftarrow$  (n1 + n2) / 2;
    se (media  $\geq$  6)
    {
        imprima ("Aprovado");
    }
    senão
    {
        imprima ("Reprovado");
    }
}
```

# Algoritmos

## Formas de representação

- Nesta disciplina não é exigida nenhuma forma específica de representação de algoritmos. De forma geral, você deverá resolver todos os exercícios na linguagem C#.
- No entanto, descrever o algoritmo antes de escrever o código em C# muitas vezes auxilia o aluno a compreender melhor o problema e a solução acaba sendo mais fácil.



# Algoritmos - exercícios

1. Utilizando um jarro de 5 litros, um jarro de 3 litros (sem quaisquer marcações de capacidade) e uma fonte de água, elabore um algoritmo para obter exatamente 4 litros d'água.
2. Você recebe 8 esferas de tamanho idêntico, sendo que apenas uma delas tem peso superior às outras. Utilizando uma balança de dois pratos, faça um algoritmo para encontrar a esfera com peso diferente sabendo que a balança só pode ser usada duas vezes.

# Introdução e Conceitos Básicos

Aula de Exercícios

**Definição:** sequência finita de instruções que, quando executada, resolve o problema proposto.

- Um algoritmo pode ser representado de diversas formas (descrição narrativa, fluxograma, diagrama de Chapin, pseudocódigo).

- “É um conjunto de regras sintáticas e semânticas usadas para definir um programa de computador. Uma linguagem permite que um programador especifique precisamente sobre quais dados um computador vai atuar, como estes dados serão armazenados ou transmitidos e quais ações devem ser tomadas sob várias circunstâncias.”

- A primeira linguagem de programação surgiu na década de 50.
- Lista de 2500 (!) linguagens de programação  
<http://people.ku.edu/~nkinners/LangList/Extras/langlist.htm>
- Linha do tempo com aproximadamente 50 linguagens de programação  
<http://www.levenez.com/lang/lang.pdf>

- Linguagem de programação que será utilizada durante a disciplina.
- C# é uma linguagem de programação orientada a objetos que é simples e poderosa. Mantém a expressividade e a elegância do estilo de linguagens C e acrescenta inovações que permitem o desenvolvimento rápido de aplicações.



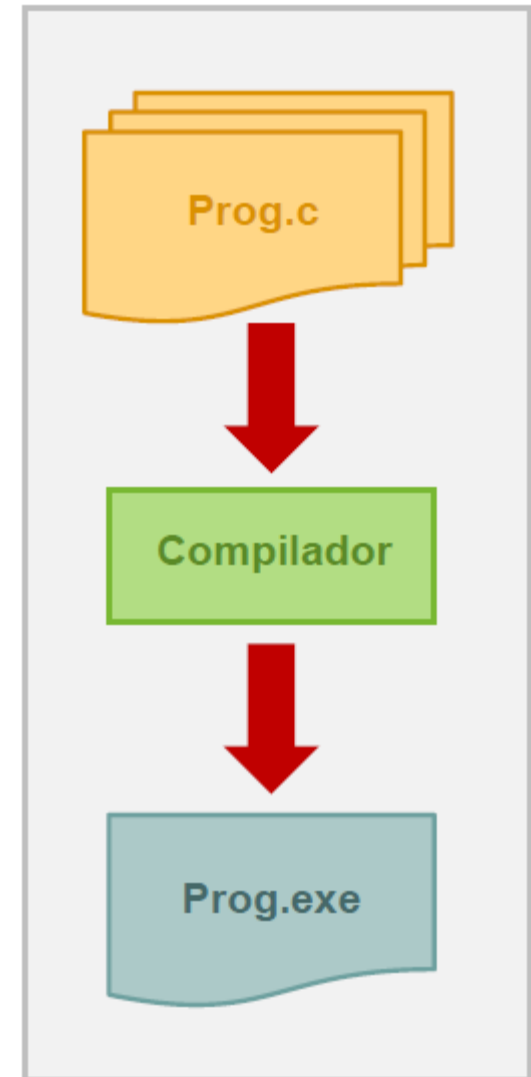
- Um programa é um algoritmo escrito em uma linguagem de programação.
- Programas devem ter uma finalidade específica:
  - Games
  - Processadores de Texto
  - Navegadores (*Browsers*)
  - Etc.
- Para que um programa escrito em *Linguagem de Programação* seja executado no computador, é necessário anteriormente convertê-lo para *Linguagem de Máquina* (0's e 1's).

# O compilador

O Programador escreve um programa em uma **Linguagem de Programação** (por exemplo, a linguagem C).

O compilador é um programa específico utilizado para traduzir as instruções definidas em C para Linguagem de Máquina.

O resultado é o arquivo executável do programa em Linguagem de Máquina.



# O compilador

- O compilador também é um programa, escrito em alguma linguagem de programação.
- Quem controla a execução do compilador (e de qualquer outro programa) é o Sistema Operacional do computador.

# Sistema Operacional (SO)

- É um programa especial que controla e coordena todas as operações básicas de um computador.
- Ele controla a execução de outros programas e pode proporcionar funções como:
  - controle de entrada e saída de dados;
  - alocação de memória;
  - gerenciamento de dados, etc ...

# IDE – *Integrated Development Environment*

- **Ambiente Integrado de Desenvolvimento:**  
programa de computador que reúne características e ferramentas de apoio ao desenvolvimento de *software* com o objetivo de agilizar este processo.

# IDE – *Integrated Development Environment*

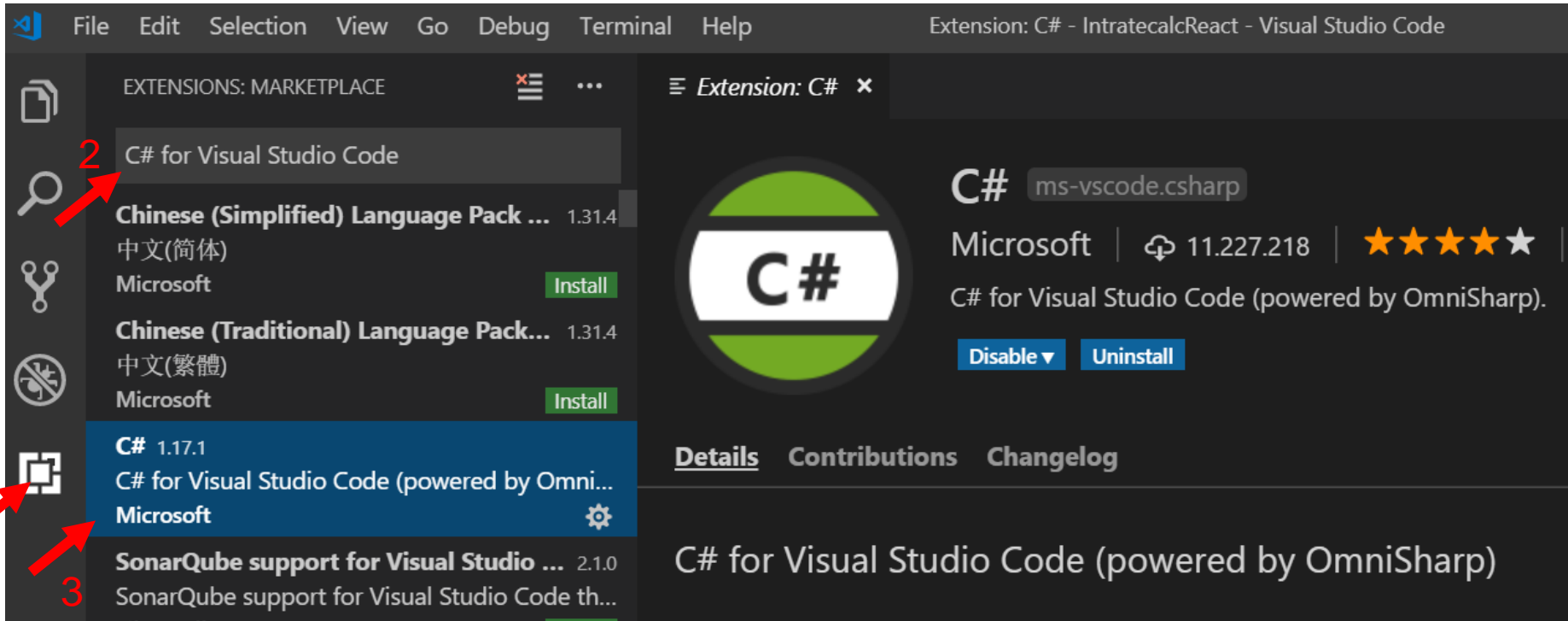
- Ambiente para programação que combina:
  - editor de texto (especificamente desenvolvido para edição de código fonte),
  - compilador,
  - depurador (funcionalidade que ajuda a testar e encontrar erros em um código),
  - entre outros.



- Visual Studio Code (VS Code) - IDE gratuita utilizada durante a disciplina.
- Download disponível em <https://code.visualstudio.com/download/>

- Extensão: C#  
(ms-vscode.csharp)

C# for Visual Studio Code (powered  
by OmniSharp)



File Edit Selection View Go Debug Terminal Help

Extension: C# - IntratecalcReact - Visual Studio Code

EXTENSIONS: MARKETPLACE

1

2

3

C# for Visual Studio Code

Chinese (Simplified) Language Pack ... 1.31.4  
中文(简体)  
Microsoft Install

Chinese (Traditional) Language Pack... 1.31.4  
中文(繁體)  
Microsoft Install

C# 1.17.1  
C# for Visual Studio Code (powered by Omni...  
Microsoft

SonarQube support for Visual Studio ... 2.1.0  
SonarQube support for Visual Studio Code th...

C# ms-vscode.csharp

Microsoft | 11.227.218 | ★★★★★

C# for Visual Studio Code (powered by OmniSharp).

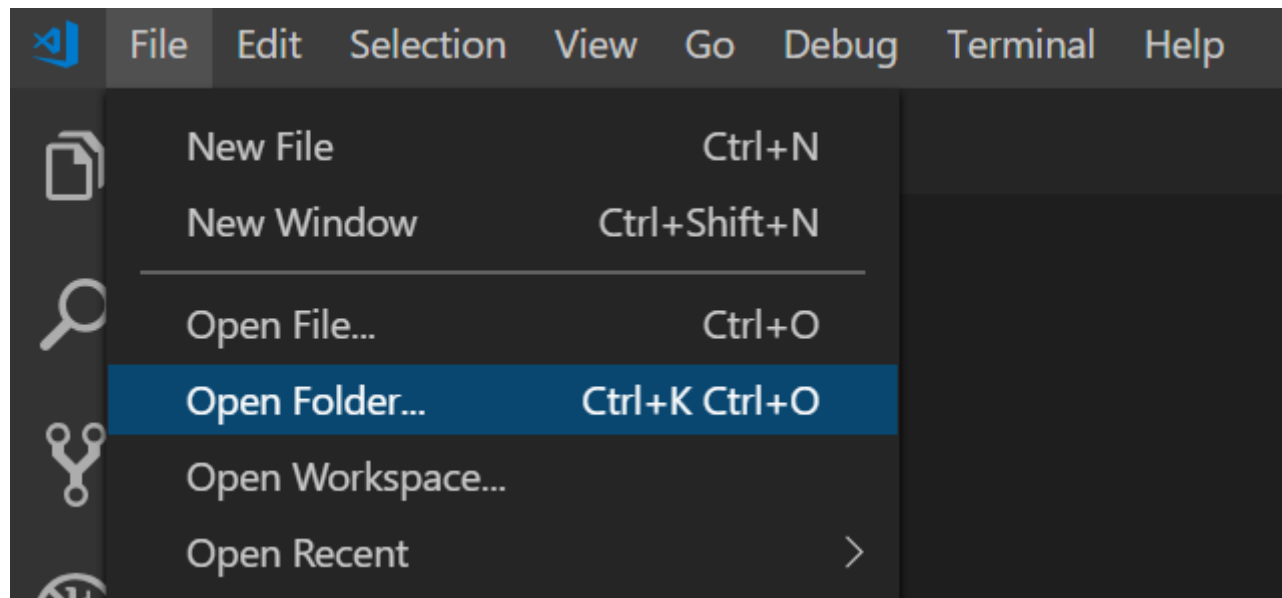
Disable Uninstall

Details Contributions Changelog

C# for Visual Studio Code (powered by OmniSharp)

## Criando um programa:

1. Crie uma pasta no Windows Explorer para armazenar o seu programa.
2. Abra a pasta criada no VS Code:



Criando um programa:

2. Pressione **CTRL** + “

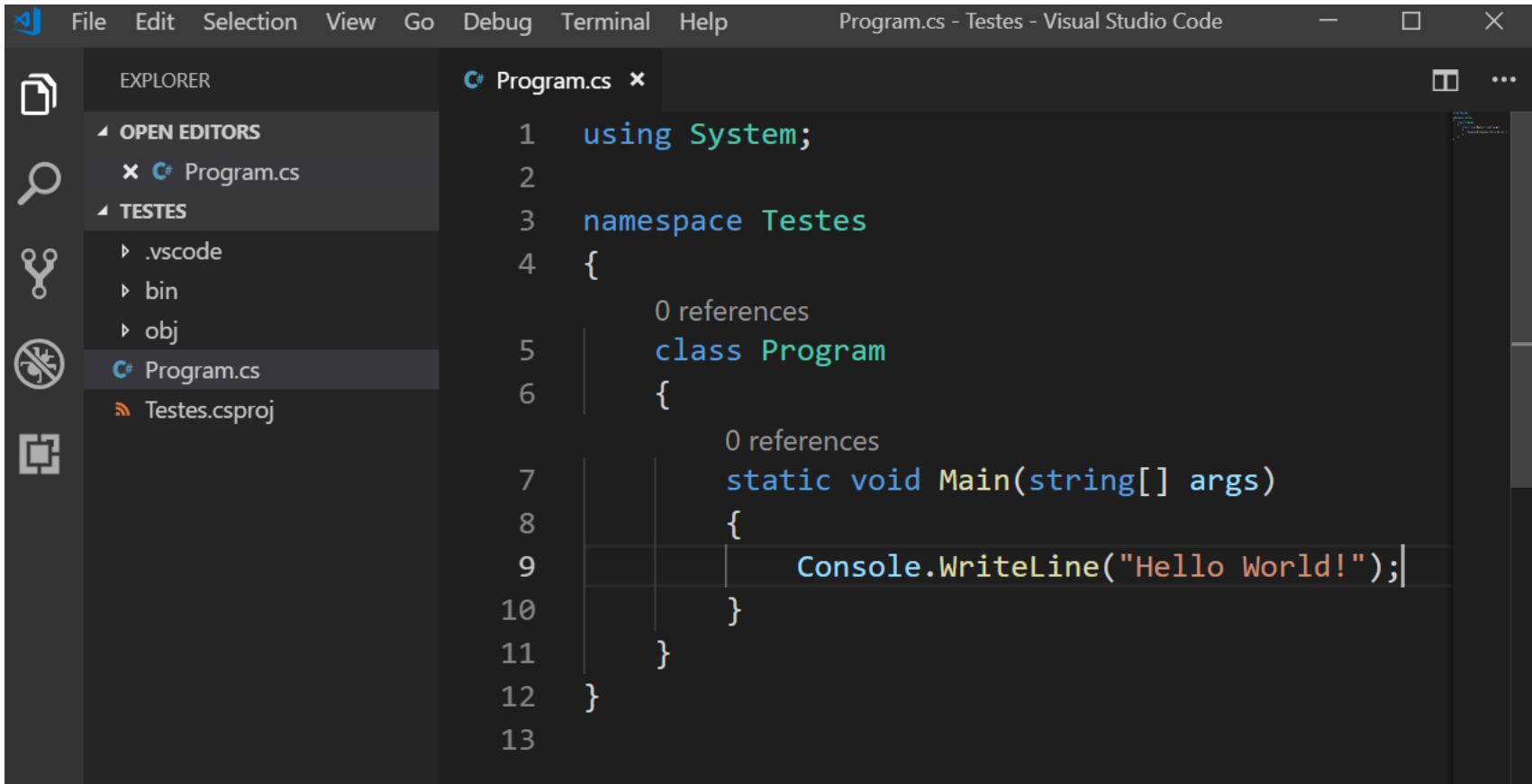
3. No Terminal, digite:

**dotnet new console**

4. Após a restauração, para rodar o programa, digite:

**dotnet run**

Duplo-clique em Program.cs apresenta um programa simples em C#:



```
1  using System;
2
3  namespace Testes
4  {
5      0 references
6      class Program
7      {
8          0 references
9          static void Main(string[] args)
10         {
11             Console.WriteLine("Hello World!");
12         }
13     }
```



- Toda linguagem de programação deve seguir uma sintaxe.
- A sintaxe são regras detalhadas que permitem a construção de uma instrução válida.
- Como ainda não aprendemos nada sobre a sintaxe da linguagem C#, nesta aula vamos ver apenas exemplos simples.

- Neste começo vamos utilizar sempre uma estrutura básica para construir nossos programas. Esta estrutura não será explicada agora, mas ao longo da disciplina você irá entendê-la melhor.

```
using System;

namespace NomeDoProjeto
{
    class Program
    {
        public static void Main(string[] args)
        {

        }
    }
}
```

- A classe Program possui o método Main que é o primeiro método do programa a ser executado. Para o código abaixo, nada é feito.

```
using System;

namespace NomeDoProjeto
{
    class Program
    {
        public static void Main(string[] args)
        {

        }

    }
}
```

- Para seu primeiro programa fazer alguma coisa, use a instrução `Console.WriteLine` para imprimir um texto na tela, como no exemplo abaixo. Rode o programa e note que uma tela preta apenas piscou em sua tela.

```
using System;

namespace NomeDoProjeto
{
    class Program
    {
        public static void Main(string[] args)
        {
            Console.WriteLine("Olá mundo!");
        }
    }
}
```

1. Faça um programa que imprima o seu nome e sobrenome no formato (Sobrenome, Nome).
2. Sabendo que a instrução `Console.WriteLine("")` imprime o texto que estiver entre aspas e salta a linha, faça um programa que escreva a inicial de seu nome utilizando sequências de espaços e letras X. Exemplo:

```
xxx      xxx
xxxx     xxxx
xx xx  xx xx
xx  xxx  xx
xx      x  xx
```
3. Teste a seguinte instrução:  
`Console.WriteLine("Teste\nde\nquebra\nde\nlinha");`  
Entendendo que `\n` faz com que o programa salte uma linha, reescreva o programa feito no exercício 2 de forma que o mesmo utilize somente um comando do tipo `Console.WriteLine` e um comando `Console.ReadKey`.