Algoritmos e Programas

Profª. Rose Yuri Shimizu

Programação

- Início da programação
  - Programas eram feitos direto nos circuitos
  - Cada novo programa: novo circuitos
- Hardware de uso geral
  - Máquina programável
  - Projetado com um limitado conjunto de instruções
  - Cada instrução habilita um circuito digital responsável por executar uma tarefa
- Programas
  - Sequência de instruções de máquina
  - Logicamente organizados
  - Codificado em uma **linguagem formal de programação**

Algoritmos x Programas

- Método para resolver **problemas** possíveis de serem implementados como um programa de computador

- Em Ciência da Computação:
  1. Sequência de **instruções** (passos):
     - ★ **Ordenada**,
     - ★ **Finita**,
     - ★ **Não ambígua**,
  2. Que produz, em tempo finito, a **solução do problema**;

- **Independe da linguagem de programação**
  - ▶ Um programa em uma linguagem é somente uma forma de implementar um algoritmo

- Garante soluções para problemas recorrentes.

Programas e linguagens

- **Programas**
  - Sequência de instruções de máquina
  - Logicamente organizados
  - Codificado em uma **linguagem formal de programação**
- Linguagem de programação
  - Máquina: conjunto de instruções implementados em hardware (binário)
  - Assembly: mnemônico ("palavra-chave") que correlaciona a funcionalidade e a instrução de máquina
  - Alto nível: "idioma", próximo à linguagem natural, que pode ser traduzido para o código de máquina

Linguagem de Máquina

- Mais próximos da linguagem interpretada pelo processador e mais distante das linguagens naturais
- **Conjunto de instruções** em hardware:
  - Operações básicas projetado em circuitos eletrônicos que não mudam
  - Define tudo o que o computador sabe fazer
  - Cada instrução possui um **código que o identifica (sequência de bits - palavra binária)** e **habilita um circuito distinto**
  - Contém instruções elementares, exemplo hipotético
    - ★ 1011000000000001 significa colocar valor 1 no registrador;
    - ★ 1111111011000000 incrementa de 1 o valor do registrador.

Linguagem de Máquina

- A associação entre os **códigos binários e a operação** (micro-programas) estão gravados em ROM
- Cada fabricante define o seu conjunto de instruções (arquitetura: MIPS, ARM, x86)
- Operações comuns:
  - Lógicas/aritméticas (soma, subtração, and, or)
  - Transferência de dados (acesso memória)
  - Controle de fluxo (desvio de código - acesso a diferentes porções da memória)
  - Interface com a E/S (acesso aos dados dos dispositivos)
- Programação impraticável para escrita e leitura

Linguagem Assembly

- Versão "legível" da linguagem de máquina
- Uma linguagem simbólica: utiliza palavras abreviadas (mnemônicos) indicando a operação "MOV R1, R2", "ADD R1, R2"

C

```
int V[NNN];
...
V[0] = V[1] + V[2]*16;
```

assembly

```
la   r1, V      # r1 <- &(V[0])
lw   r4, 4(r1)  # r4 <- M[r1+1*4]
lw   r6, 8(r1)  # r6 <- M[r1+2*4]
sll  r6, r6, 4  # r6*16 = r6<<4
add  r7, r4, r6
sw   r7, 0(r1)  # M[r1+0*4] <- r4+r6
```

- Vantagens: otimização, maior controle sobre os recursos do computador
- Desvantagem: exige conhecimento sobre a arquitetura e organização
- Montador (assembler): Programa Assembly $\rightarrow$ Linguagem de Máquina.

Linguagem de programação de alto nível

- É um "idioma" que pode ser **traduzido para o binário**
- **Sintaxe mais próxima da linguagem natural**
- Possui conjunto limitado de palavras (*vocabulário*)
- Associado a um conjunto de regras (*sintaxe*):
  - Define quais são as palavras válidas da LP (palavras reservadas, identificadores, operadores, etc) - *análise léxica*
  - Define quais são as combinações de palavras formam instruções válidas (comandos, expressões, subrotinas, etc) - *análise sintática*
- Programas:
  - Sequências de instruções escritas em uma linguagem de programação
  - Código-fonte: arquivo textual que contém o programa
  - Código-objeto: arquivo binário que contém as instruções de máquina do processador
  - Sistema operacional: responsável por enviar as instruções
  - Processador: responsável por executar (ativar circuitos/dispositivos)

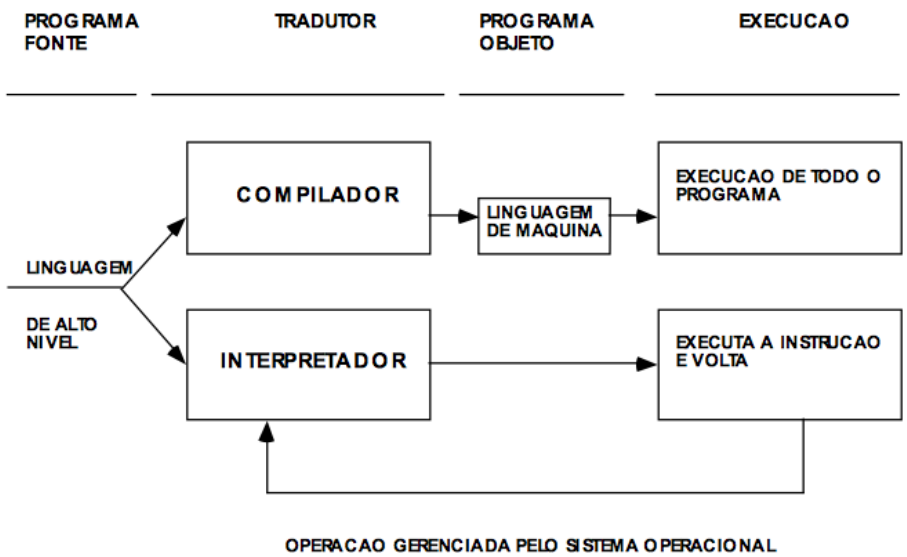
Linguagem de alto nível - Compiladas

- Compilador (programa) converte o código-fonte em código-objeto
- Dependentes de plataforma: código gerado é para rodar em uma plataforma específica (Windows, Linux, MacOS)
- Vantagem: mais rápido
- Executar o programa: "rodar/chamar" o código-objeto
  1. Escrever código-fonte (exemplo.c)
  2. Compilar: gcc exemplo.c -o exemplo
  3. Executar: ./exemplo
- Só é necessário recompilar o programa caso haja mudanças no código-fonte
- Linguagem compilada: C, C++, C#, Java (gera um bytecode), Rust

Linguagem de alto nível - Interpretadas

- Interpretador (programa) traduz cada instrução convertendo para instrução de máquina
- Vantagem:
  - Código-fonte: independente de plataforma
  - Intepretador: dependente de plataforma
- Executar o programa: "rodar/chamar" intepretador para executar o código-fonte
  1. Escrever código-fonte (exemplo.py)
  2. Executar: python exemplo.py
- Linguagem interpretada: PHP, Javascript, Python

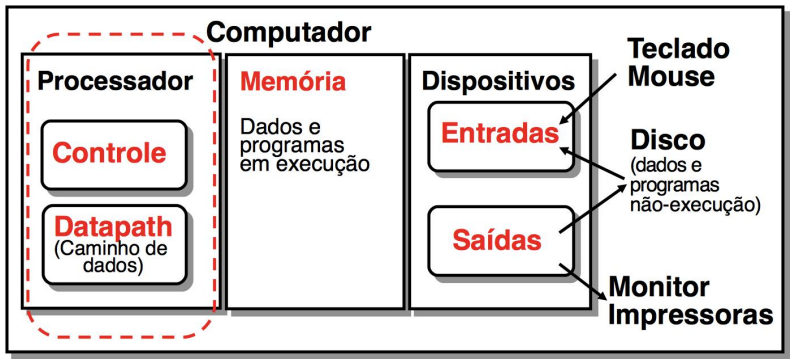
Compilador x Interpretador



- Compilado: "conversa" diretamente com um nativo
- Interpretado: intérprete traduz a "conversa" para um nativo

Execução de um programa

- **Armazenamento**: memória não-volátil(ex. HD)
- **Execução**:
  - Sistema operacional carrega para a memória principal (RAM)
  - Acessível pela CPU:
    - ★ Ciclo de instrução: busca (RAM), decodificação (UC - unidade de controle), execução (ULA - unidade lógica e aritmética)
  - Um programa em execução é chamado de processo (gerenciado pelo SO)



Processo de compilação

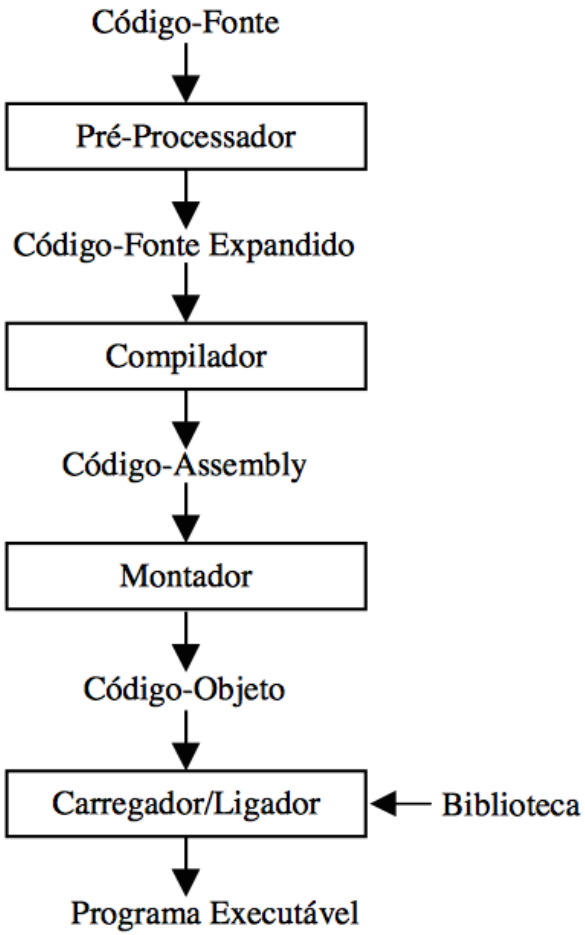
- Tradução do Código Fonte (código objeto)
- Linguagem de Alto Nível $\rightarrow$ Linguagem de Máquina;
- Código-fonte:
  - Análise léxica (palavras) e sintática (estrutura gramatical): de acordo com a gramática da linguagem utilizada
  - Semântica: faz sentido - se pode ser executado

```
int main()
{
    i = 10;
    return 0;
}
```

Sintaxe sim
Semântica não

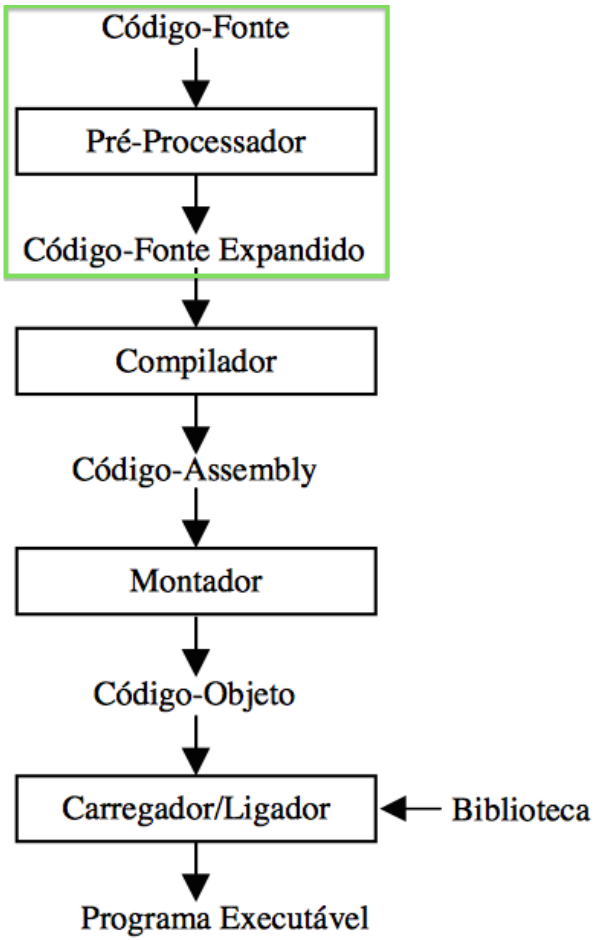
Processo de compilação

Geração do código de máquina



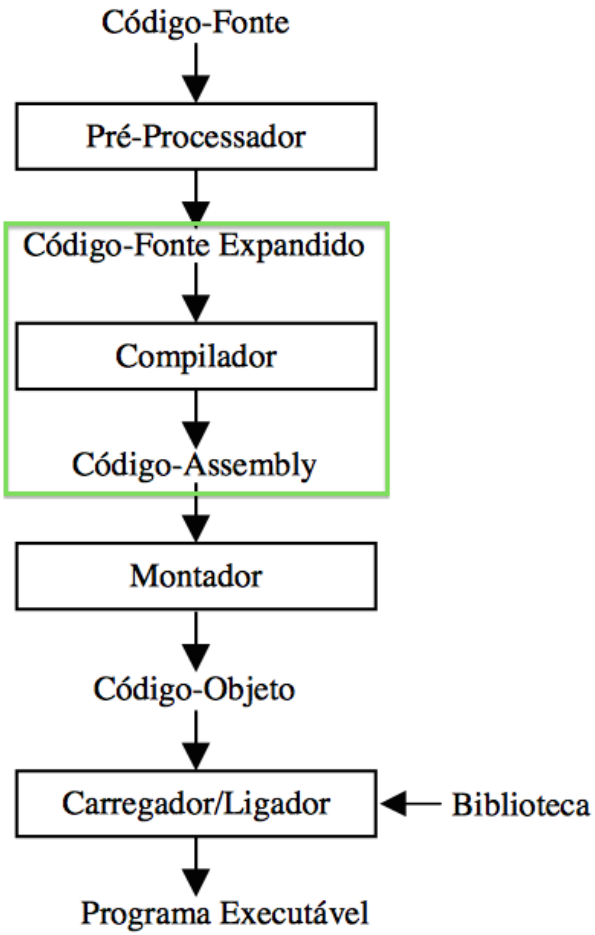
1. **Programa**: sequência de instruções em linguagem de alto nível;
2. **Linguagem alto nível**: compreensível aos humanos (int a = 1 + 5);
3. **Processo de compilação**: geração do código de máquina;
4. **Programa executável** pelo computador.

Processo de compilação



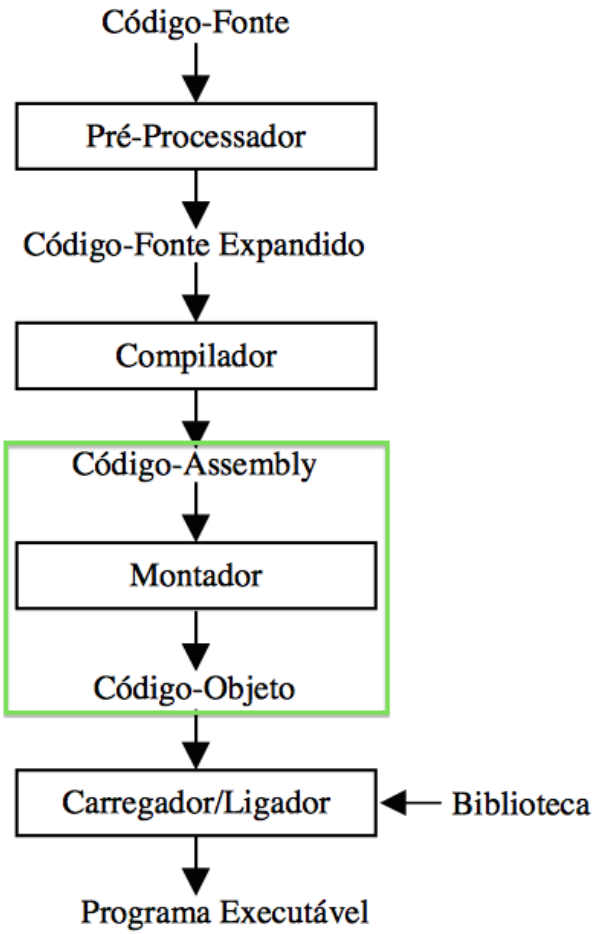
- **Pré-processador**:
  - Processa informações (substituições e inclusões no código fonte) que serão utilizadas pelo programa.

Processo de compilação

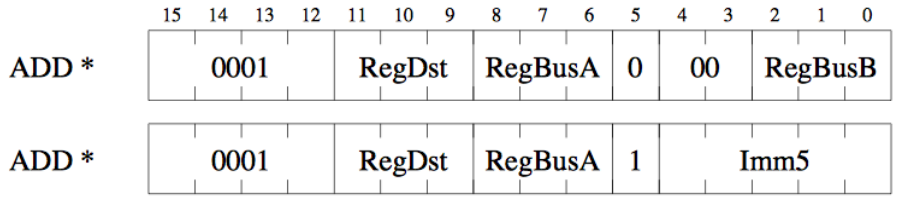


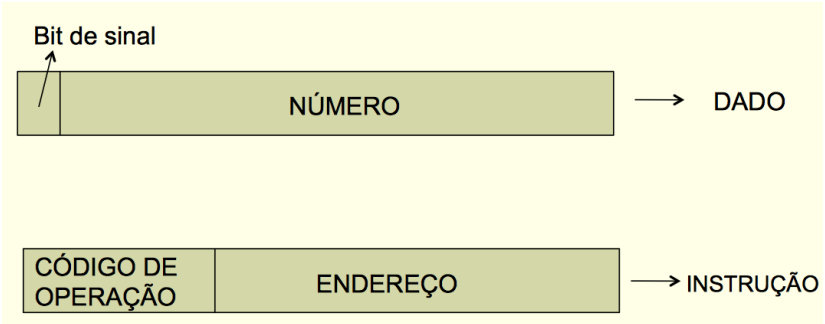
- **Pré-processador**: substituições e inclusões no código fonte;
- **Compilador**:
  - Análise léxica, sintática e semântica;
  - Gera o código de montagem assembly:
    - Códigos mnemônicos que correspondem aos códigos de máquina ('read', 'add', 'load', etc).
    - int a = b + c $\rightarrow$ add $r3,$r1,$r2 (assembly).

Processo de compilação



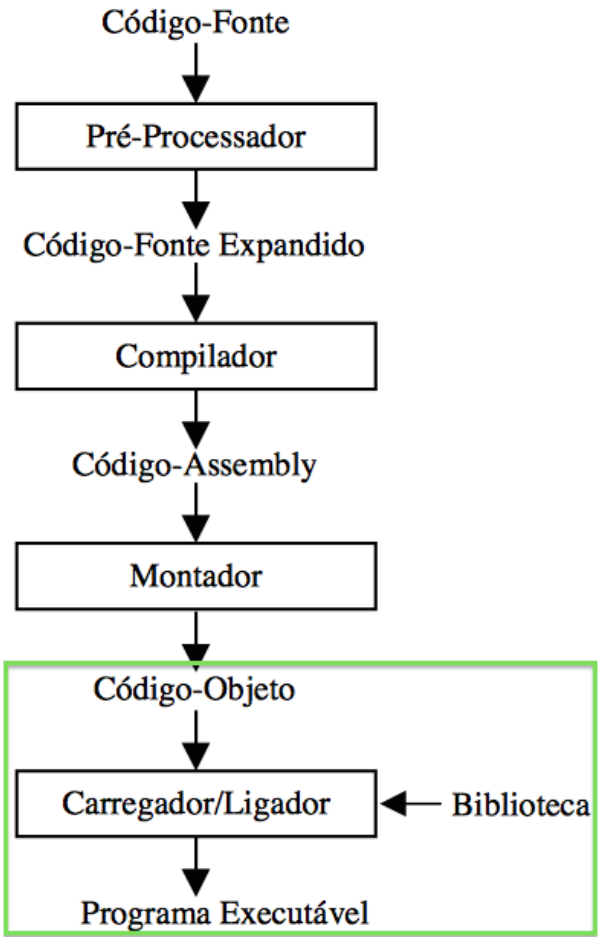
- **Pré-processador**: substituições e inclusões no código fonte;
- **Compilador**: gera o código assembly;
- **Montador**: gera o código de máquina;





- int a = b + c
- add $r3,$r1,$r2 (assembly)
- 0001 010 101 0 00 100

Processo de compilação



- **Pré-processador**: substituições e inclusões no código fonte;
- **Compilador**: código assembly;
- **Montador**: código de máquina;
- **Linker**:
  - Ligações no código de máquina;
  - Liga as bibliotecas e códigos objetos necessários para gerar o executável.

Algoritmos: narrativa

- Consiste em analisar o enunciado do problema e escrever, utilizando uma linguagem natural (português), seguindo passos para a sua resolução;

- Algoritmo para saber se pode comprar ou não um produto de acordo com o saldo bancário e preço do produto
  1. **Entrada dos dados** - Receber o saldo e o preço
  2. **Processamento dos dados** $\rightarrow$ **informação** - Se o saldo é maior ou igual ao preço do produto
  3. **Saída das informações** - O produto pode ser comprado

Escreva um algoritmo para verificar se passou na disciplina.

1. Receba os dados (**entrada**):

Escreva um algoritmo para verificar se passou na disciplina.

1. Receba os dados (**entrada**):
   - frequência (F), notas da prova 1 (P1), da prova 2 (P2), da prova 3 (P3) e prova 4(P4)

Escreva um algoritmo para verificar se passou na disciplina.

1. Receba os dados (**entrada**):
   - frequência (F), notas da prova 1 (P1), da prova 2 (P2), da prova 3 (P3) e prova 4(P4)
2. Calcular a média (**processamento**):

Escreva um algoritmo para verificar se passou na disciplina.

1. Receba os dados (**entrada**):
   - frequência (F), notas da prova 1 (P1), da prova 2 (P2), da prova 3 (P3) e prova 4(P4)
2. Calcular a média (**processamento**):
   - $NF = \frac{(P1 + (2 * P2) + (3 * P3))}{6}$

Escreva um algoritmo para verificar se passou na disciplina.

1. Receba os dados (**entrada**):
   - frequência (F), notas da prova 1 (P1), da prova 2 (P2), da prova 3 (P3) e prova 4(P4)
2. Calcular a média (**processamento**):
   - $NF = \frac{(P1 + (2 * P2) + (3 * P3))}{6}$
3. Verificação (**processamento**):

Escreva um algoritmo para verificar se passou na disciplina.

1. Receba os dados (**entrada**):
   - frequência (F), notas da prova 1 (P1), da prova 2 (P2), da prova 3 (P3) e prova 4(P4)
2. Calcular a média (**processamento**):
   - $NF = \frac{(P1 + (2 * P2) + (3 * P3))}{6}$
3. Verificação (**processamento**):
   1. Se a $NF \geq 6$ e $F \geq 75$,

Escreva um algoritmo para verificar se passou na disciplina.

1. Receba os dados (**entrada**):
   - frequência (F), notas da prova 1 (P1), da prova 2 (P2), da prova 3 (P3) e prova 4(P4)
2. Calcular a média (**processamento**):
   - $NF = \frac{(P1 + (2 * P2) + (3 * P3))}{6}$
3. Verificação (**processamento**):
   1. Se a $NF \geq 6$ e $F \geq 75$,
      - ★ está aprovado, (**saída**)

Escreva um algoritmo para verificar se passou na disciplina.

1. Receba os dados (**entrada**):
   - frequência (F), notas da prova 1 (P1), da prova 2 (P2), da prova 3 (P3) e prova 4(P4)
2. Calcular a média (**processamento**):
   - $NF = \frac{(P1 + (2 * P2) + (3 * P3))}{6}$
3. Verificação (**processamento**):
   1. Se a $NF \geq 6$ e $F \geq 75$,
      - ⋆ está aprovado, (**saída**)
   2. Senão,

Escreva um algoritmo para verificar se passou na disciplina.

1. Receba os dados (**entrada**):
   - frequência (F), notas da prova 1 (P1), da prova 2 (P2), da prova 3 (P3) e prova 4(P4)
2. Calcular a média (**processamento**):
   - $NF = \frac{(P1 + (2 * P2) + (3 * P3))}{6}$
3. Verificação (**processamento**):
   1. Se a $NF \geq 6$ e $F \geq 75$,
      - ⋆ está aprovado, (**saída**)
   2. Senão,
      - ⋆ está reprovado (**saída**)

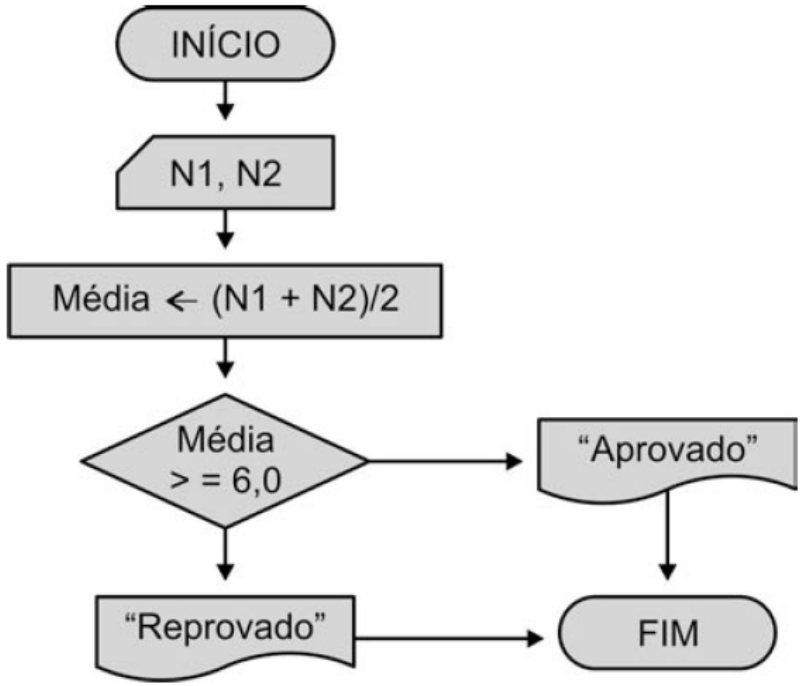
Algoritmos: fluxograma

- Utilização de símbolos gráficos predefinidos para representar os passos a serem seguidos para sua resolução.

Símbolo	Descrição
	Símbolo utilizado para indicar o início e o fim do algoritmo.
	Símbolo que permite indicar o sentido do fluxo de dados. Serve exclusivamente para conectar os símbolos ou blocos existentes.
	Símbolo utilizado indicar cálculos e atribuições de valores.
	Símbolo utilizado para representar a entrada de dados.
	Símbolo utilizado para representar a saída de dados.
	Símbolo utilizado para indicar que deve ser tomada uma decisão, apontando a possibilidade de desvios.

Algoritmos: fluxograma

- Algoritmo para calcular a média aritmética entre duas notas de um aluno e mostrar sua situação, que pode ser aprovado ou reprovado.



Algoritmos: pseudocódigo

- Forma genérica de escrever um algoritmo
- Utiliza uma linguagem simples (portugol)
- Não pode ser executado em um sistema real (computador)
- É uma linguagem de **projeto de programação**:
  - Facilita o **raciocínio** lógico do programa
  - E assemelha-se à linguagem formal de programação
- Elementos recorrentes: variáveis, tipos de dados, estrutura de controle

```
1 programa primeiro
2 inicio
3     inteiro x;
4     x = 29;
5     escreva (x);
6 fim
```

Pseudocódigo - Exemplo

- Algoritmo para calcular a média aritmética entre duas notas de um aluno e mostrar sua situação, que pode ser aprovado ou reprovado.

```
1  programa media
2  inicio
3      real nota1,nota2,media;
4
5      leia (nota1, nota2);
6      media = (nota1 + nota2)/2;
7      se (media <= 7)
8      inicio
9          escreva ("Reprovado");
10     fim
11     senao
12     inicio
13         escreva ("Aprovado");
14     fim
15 fim
16
```

Algoritmos x Programa

- Algoritmo: independe da linguagem de programação
- Programa: algoritmo implementado em uma linguagem formal de programação
- Algoritmo $\rightarrow$ Linguagem de programação $\rightarrow$ Programa
- Linguagem formal utilizada na disciplina: linguagem C

Linguagem de programação C

- Linguagem C nasceu na década de 70
- Inventado por Dennis Ritchie, reescreveu o sistema operacional UNIX
  - Originalmente UNIX foi implementado em assembly
- Derivado da linguagem B, criado por Ken Thompson
- Linguagem propósito geral
  - Implementa instrução da arquitetura (linguagem de máquina)
  - Desenvolvimento de programas diversos como processadores de texto, planilhas eletrônicas, sistemas operacionais, programas para a automação industrial, gerenciadores de bancos de dados, etc.

Linguagem de programação C

- Compilada
  - Possui conjunto limitado de palavras (*vocabulário*)
  - Associado a um conjunto de regras (*sintaxe*)
    - Palavras válidas: palavras reservadas, identificadores, operadores, etc - *análise léxica*
    - Combinações de palavras formam instruções válidas (comandos, expressões, funções, etc) - *análise sintática*
  - Traduzida para linguagem de máquina pelo compilador
- Paradigma:
  - Imperativa: ordens através de instruções
  - Estruturada
    - Sequencial: a tarefa é executada logo após a outra;
    - Condicional (decisão): a tarefa é executada após um teste lógico, e;
    - Iterativa (repetição): a partir do teste lógico, um trecho do código pode ser repetido finitas vezes.
  - Procedural: blocos/trechos de códigos com função específicas, que podem ser reutilizados

Programas C - Estrutura básica

- Ambiente:
  - Abra o editor: tecla Windows () + digite “editor” + clique no “Text Editor”
    - ★ Ctrl + s (salvar)
    - ★ Selecione o diretório “Documentos”
    - ★ Salvar como “programa.c”
- Primeiro programa:

```c
#include <stdio.h>

int main() {




    return 0;
}
```

```c
//"Bibiblioteca padrão de entrada/saída" do C (libc) - C11/C17/C23
#include <stdio.h>

//Procedural: blocos/trechos de códigos com função específicas
//            blocos marcados com as chaves { }

//main: função principal é a primeira instrução executada
//      início da execução
int main()
{
    //Imperativa: instruções comandam o que deve ser executado
    //Estruturada: as instruções são executadas sequencialmente
    //             porém podem haver desvios nessa ordem


    return 0; //padrao C: return 0 sucesso e não-zero erro
              //exit(0);
              //EXIT_SUCCESS;

}
```

Função **printf()** - saída

```c
#include <stdio.h>

int main()
{
    //Imprime mensagens na tela
    //Imprime tudo que esta entre aspas dupla
    //Toda instrução termina com ";"
    //\n nova linha e fim de linha

    printf("Alo mundo\n");

    printf("Alo");
    printf(" mundo\n");

    printf("Alo"
           " mundo\n");

    printf("Alo");
    printf(" ");
    printf("mundo\n");

    return 0;
}
```

Função **printf()** - saída

- Abra o terminal: tecla Windows () + digite "terminal" + clique no "Terminal"
- Verifique se o diretório "Documentos" encontra-se no diretório atual: ls
- Entre no diretório "Documentos": cd Documentos
- Liste os arquivos que estão dentro do diretório "Documentos": ls
- Compilar
  - **gcc programa.c -o programa**
  - gcc é o compilador:
    - Tradução do Código Fonte → código objeto
    - Linguagem de Alto Nível → Linguagem de Máquina
    - Efetua a análise léxica (palavras) e sintática (estrutura gramatical) e análise semântica (faz sentido - se pode ser executado) do código-fonte
    - Comentários são ignorados (// /* */)
  - Ao listar o conteúdo do diretório (ls), observe que o arquivo **programa** foi gerado
- Executar
  - **./programa**
- Assembly:
  - gcc programa.c -S
  - cat programa.s