ipt Instituto Politécnico de Tomar

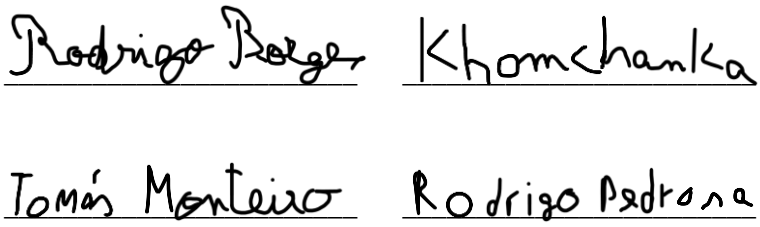
Arquitetura de Computadores

Divisão e Raiz Quadrada

Índice

Relatório realizado por Tomás Monteiro n27233, Rodrigo Borges n26256, Kanstantsin Khomchanka n27230, Rodrigo Pedrosa n27470 e Diogo Godinho n27220

Assinaturas digitais:



Introdução

Este relatório analisa dois algoritmos de divisão distintos, apresentados sob a forma de fluxogramas. O primeiro descreve o método para a **Divisão Inteira**, um processo fundamental na aritmética para encontrar um quociente e um resto. O segundo detalha um método para o **cálculo da Raiz Quadrada**, que, curiosamente, partilha semelhanças processuais com a divisão longa.

---

1. Análise do Fluxograma: Divisão Inteira

Este algoritmo implementa o método da "divisão longa" manual para encontrar o quociente e o resto da divisão de um Dividendo por um Divisor.

Etapas do Processo:

1. **Entrada e Verificação:**
    - O algoritmo começa por ler os dois valores: Dividendo e Divisor.
    - É feita uma verificação crítica: se o Divisor for igual a 0, o programa reporta um erro, uma vez que a divisão por zero é indefinida.

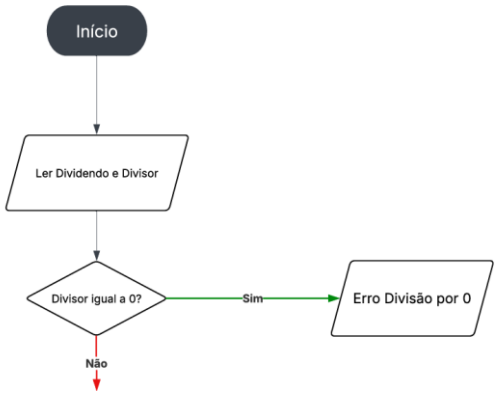


Figura 1 Entrada e Verificação

2. **Inicialização:**
    - Se o divisor for válido, as variáveis de resultado são inicializadas: Resto = 0 e Quociente = 0.
    - O algoritmo determina a qntD (Quantidade de Dígitos) do dividendo, que controlará o ciclo principal.

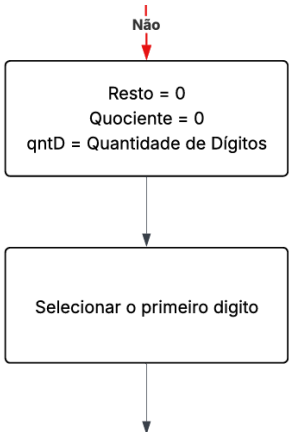


Figura 2 Inicialização

3. **Ciclo Principal (Processamento por Dígito):**
    - O algoritmo entra num ciclo que se repete enquanto qntD > 0.
    - **"Baixar" o dígito:** O Resto da iteração anterior é multiplicado por 10 e somado ao "dígito atual" do dividendo. Isto é análogo a "baixar" o próximo algarismo na divisão manual.
    - **Ciclo Interno (Encontrar 'q'):** Inicia-se um sub-processo para encontrar quantas vezes o Divisor "cabe" no Resto atual.

- Uma variável temporária q é iniciada em 0.
- Enquanto o produto (q * Divisor) for menor ou igual ao Resto, o valor de q é guardado como q_valido e depois incrementado (iterar q).
- Quando (q * Divisor) ultrapassa o Resto, o ciclo interno para. O q_valido retém o maior número de vezes que o divisor coube no resto.

- **Atualização de Valores:**
  - O q_valido (o dígito encontrado para o quociente) é usado para calcular o novo Resto: Resto = Resto - (q_valido * Divisor).
  - O Quociente é atualizado "anexando" o novo dígito: Quociente = (Quociente * 10) + q_valido.
- O contador de dígitos qntD é decrementado, e o ciclo principal recomeça com o próximo dígito.

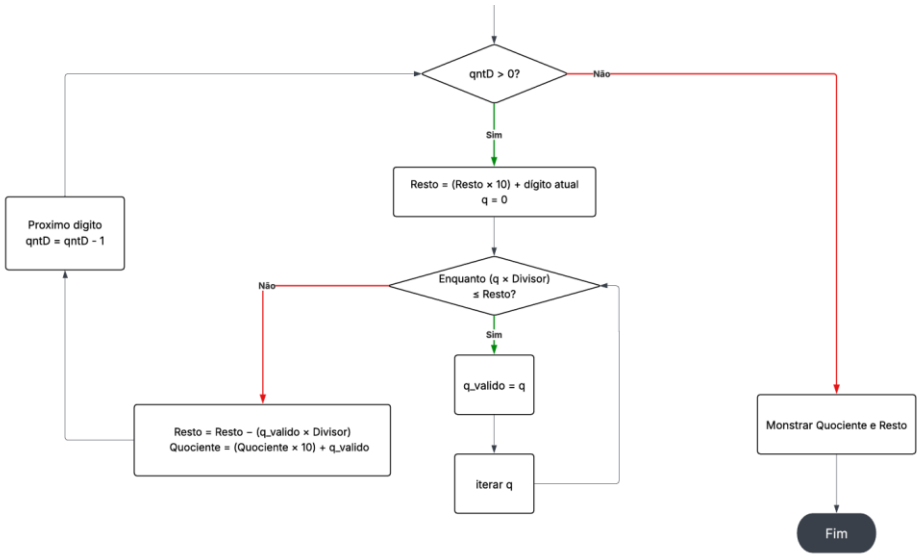


Figura 3 Ciclo Principal

4. **Resultado:**
   - Quando todos os dígitos forem processados (qntD chega a 0), o ciclo termina.

- O algoritmo mostra o Quociente e o Resto finais.

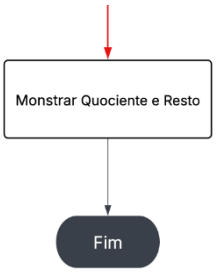


Figura 4 Resultado

---

2. Análise do Fluxograma: Raiz Quadrada

Este algoritmo calcula a raiz quadrada de um número N usando um método iterativo semelhante à divisão longa, mas que opera sobre pares de dígitos.

Etapas do Processo:

1. **Entrada e Verificações:**
   - O algoritmo lê o número N.
   - Verifica se N < 0. Se for, reporta um erro, pois não existem raízes (reais) de números negativos.
   - Verifica o caso especial N = 0, mostrando N como resultado.

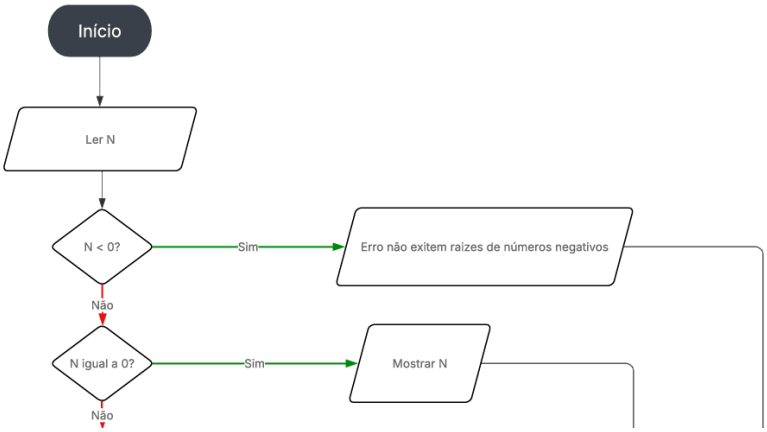


Figura 5 Entrada e Verificações

2. **Inicialização (Preparação dos Pares):**
   - Os dígitos de N são separados em pares, a partir da vírgula (ex: 123,4 torna-se 01 23 40).

- O algoritmo guarda o número de pares antes da vírgula (qntA) e o total de pares (qntPares).
- As variáveis de resultado são inicializadas: Resto = 0 e Raiz = 0.

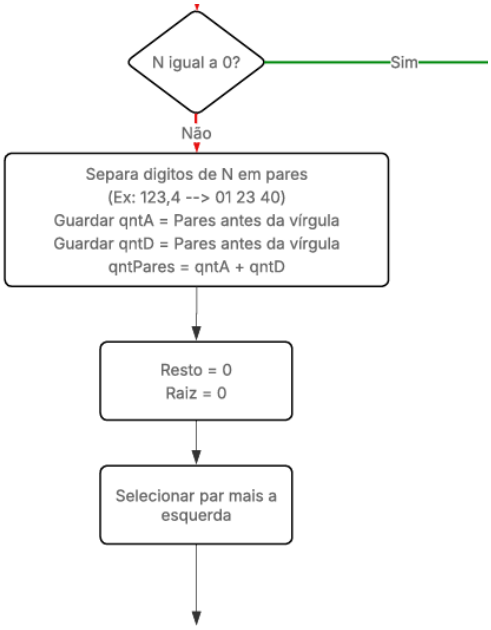


Figura 6 Inicialização

3. **Ciclo Principal (Processamento por Par):**
   - O algoritmo entra num ciclo que se repete enquanto qntPares > 0.
   - **"Baixar" o par:** O Resto é multiplicado por 100 (em vez de 10, como na divisão) e somado ao "par atual".
   - **Ciclo Interno (Encontrar 'a'):** O objetivo é encontrar o maior dígito a (de 0 a 9) que satisfaça a condição (20 * Raiz + a) * a <= Resto.
     - a é iniciado em 0.
     - O algoritmo testa a condição. Se for verdadeira (Sim), ele itera a (a torna-se a+1) e verifica se a já chegou a 9.
     - Este teste repete-se até que a condição seja falsa (Não) ou até que a chegue a 9 e passe no teste.
   - **Atualização de Valores:**
     - Uma vez que o dígito a correto é encontrado (o último que satisfez a condição), os valores são atualizados. (Nota: O

fluxograma implica que o valor a usado na atualização é o último que *passou* no teste).

- O novo Resto é calculado: Resto = Resto - (20 * Raiz + a) * a.
- A Raiz é atualizada "anexando" o dígito a: Raiz = (Raiz * 10) + a.

- O contador de pares qntPares é decrementado, e o ciclo principal recomeça com o próximo par.

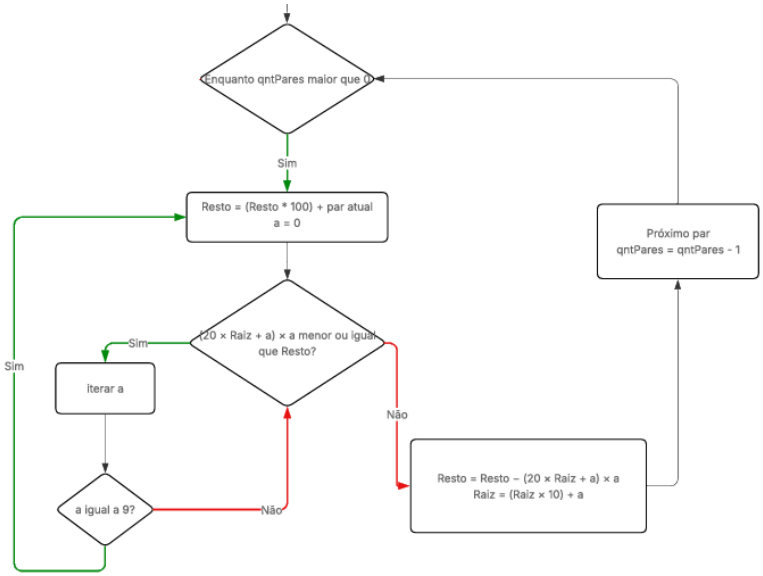


Figura 7 Ciclo Principal

4. **Ajuste Decimal e Resultado:**
   - Após o fim do ciclo, o algoritmo verifica se N tem virgula.
   - Se Sim, a vírgula é colocada na Raiz na posição correta, com base no número de pares que existiam antes da vírgula (qntA).
   - Finalmente, o algoritmo mostra a Raiz calculada e o Resto final.

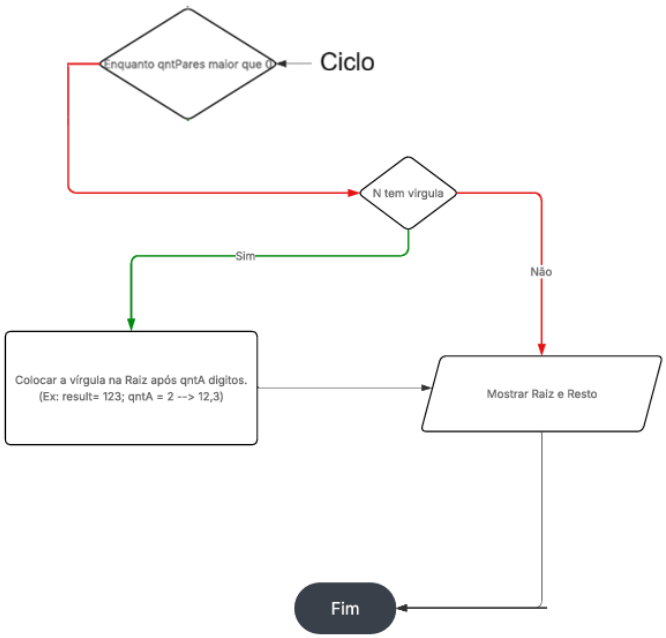


Figura 8 Ajuste Decimal e Resultado

Conclusão e Comparação

Ambos os algoritmos demonstram métodos iterativos para "desconstruir" um número e encontrar um resultado.

- **Divisão Inteira:** Processa *dígito a dígito*, atualizando o resto ao multiplicar por 10. O objetivo em cada passo é encontrar um dígito q_valido.
- **Raiz Quadrada:** Processa *par a par*, atualizando o resto ao multiplicar por 100. O objetivo em cada passo é encontrar um dígito a através de uma fórmula de teste mais complexa ((20 * Raiz + a) * a).

Apesar de calcularem resultados muito diferentes, a estrutura central de "baixar" uma porção do número, encontrar um dígito para o resultado e calcular um novo resto é conceptualmente semelhante em ambos os processos.

---

3. Pseudocódigo da Divisão Inteira

1. **Início e Entradas:**

- O processo começa e pede ao utilizador dois valores: o Dividendo e o Divisor.

```
Inicio
    Ler Dividendo
    Ler Divisor
```

Figura 9 Início e Entradas

2. Verificação de Erro (Condicional Se):
   - O algoritmo verifica imediatamente se o Divisor é igual a 0.
   - Se for, ele escreve "Erro Divisao por 0" e termina o processo principal, pois é uma operação matemática indefinida.

```
    Se Divisor = 0 Entao
        Escrever "Erro Divisao por 0"
```

Figura 10 Verificação de Erro

3. Inicialização (Bloco Senao):

   - Se o divisor for válido (diferente de 0), o algoritmo prepara as suas variáveis:
   - Resto e Quociente são iniciados em 0.
   - qntD é definida como a Quantidade de Dígitos total do Dividendo.
   - O digito_atual é definido como o primeiro dígito da esquerda do Dividendo.

```
    Senao
        Resto = 0
        Quociente = 0
        qntD = Quantidade de Digitos do Dividendo
```

Figura 11 Inicialização

4. Ciclo Principal (Primeiro Enquanto):

```
Enquanto qntD > 0 Faca
    Resto = (Resto * 10) + digito_atual
    q = 0

    Enquanto (q * Divisor) <= Resto Faca
        q_valido = q
            q = q + 1
    Fim Enquanto
```

- Este é o coração do algoritmo. Ele continuará a repetir enquanto houver dígitos para processar (qntD > 0).
- **Passo 4a: "Baixar" o dígito:** A linha Resto = (Resto * 10) + digito_atual é a representação de "baixar" o próximo algarismo. O resto anterior é multiplicado por 10 (para "abrir espaço") e o dígito atual é somado a ele.
- **Passo 4b: Iniciar o contador q:** A variável q é zerada. Esta variável vai testar quantas vezes o Divisor cabe no Resto atual.

5. Ciclo Principal (Segundo Enquanto):

- Este é o coração do algoritmo. Ele continuará a repetir enquanto houver dígitos para processar (qntD > 0).
- Este ciclo é o que de facto faz a divisão para o dígito atual.
- Ele testa se q vezes o Divisor ainda é menor ou igual ao Resto.
- Enquanto for, ele guarda o valor de q em q_valido (pois este é o último

Figura 12 Ciclo Principal (Primerio Enquanto)

valor "bom" que funcionou) e depois incrementa q (q = q + 1) para tentar o próximo número.

```
Enquanto (q * Divisor) <= Resto Faca
    q_valido = q
        q = q + 1
Fim Enquanto
```

Figura 13 Ciclo Principal (Segundo Enquanto)

```
Resto = Resto - (q_valido * Divisor)
Quociente = (Quociente * 10) + q_valido
```

6. Atualização dos Valores

   - Quando o Ciclo Interno termina (porque q * Divisor passou a ser maior que o Resto), o algoritmo usa o último q_valido que guardou:
   - Resto = Resto - (q_valido * Divisor): Calcula o novo resto, subtraindo o valor que foi efetivamente dividido.
   - Quociente = (Quociente * 10) + q_valido: Constrói o número do quociente, "anexando" o dígito q_valido à direita.

7. Preparação para a próxima Iteração

   - O algoritmo avança para o Proximo dígito do Dividendo.
   - O contador qntD é diminuído em 1 (qntD - 1), sinalizando que um dígito já foi processado.
   - O Ciclo Principal (Passo 4) recomeça.

```
        Selecionar o Proximo digito
        qntD = qntD - 1
Fim Enquanto
```

Figura 14 Preparação para a próxima iteração

8. Fim

   - Quando o Ciclo Principal termina (quando qntD chega a 0), o algoritmo sai do Fim Enquanto.
   - Ele então executa o comando final: Monstrar Quociente e Resto.
   - O Fim Se fecha o bloco de lógica e o programa termina.

```
        Monstrar Quociente e Resto
    Fim Se
Fim
```

Figura 15 Fim

4. Pseudocódigo da Raiz

1. Início e Entrada:

- O processo começa e lê o número N do qual se quer calcular a raiz.

```
Inicio
    Ler N
```

Figura 16 Início e Entrada

2. Verificações Iniciais:

- Se N < 0 Então: Primeiro, verifica se o número é negativo. Se for, escreve uma mensagem de erro, pois não se pode calcular a raiz real.
- Senao Se N = 0 Então: Trata o caso especial de N ser 0, mostrando diretamente que a raiz é 0.

```
Se N < 0 Então
    Escrever "Erro: não existem raízes de números negativos"
Senao Se N = 0 Então
    Escrever "Raiz = 0"
```

Figura 17 Verificações Iniciais

3. Inicialização e Separação de Pares):

- Este é o bloco principal para números positivos.
- O código verifica se N tem vírgula. Se não tiver, assume uma parte decimal igual a 0
- “Criar pares” Este bloco detalha o passo "Separa digitos de N em pares". De forma muito importante, ele especifica como os pares são criados: a partir da parte inteira (da direita para a esquerda) e da parte decimal (da esquerda para a direita).
- qntA é guardado como o número de pares da parte inteira (antes da vírgula).

- qntD é o número de pares da parte decimal.
- qntPares é a soma total de pares a processar.
- As variáveis de resultado são inicializadas: Resto = 0 e Raiz = 0.
- O algoritmo seleciona o primeiro par mais à esquerda para começar.

```
Senao
    Se N tem vírgula Então
        Separar parte inteira e decimal
    Senao
        parte decimal = 0
    Fim Se

    // Separar dígitos em pares
    Criar pares a partir da parte inteira (da direita para a esquerda)
    Criar pares a partir da parte decimal (da esquerda para a direita)
    qntA = número de pares da parte inteira
    qntD = número de pares da parte decimal
    qntPares = qntA + qntD

    Resto = 0
    Raiz = 0
    Selecionar par mais à esquerda
```

Figura 18 Inicialização

4. Ciclo Principal (Enquanto):

- Este é o ciclo principal, que se repete enquanto qntPares > 0.
- Resto = (Resto * 100) + parAtual: "Baixa" o próximo par de dígitos. O resto anterior é multiplicado por 100 (porque estamos a usar pares) e o par atual é somado.
- a = 0: Inicia o dígito de teste a em 0 para esta iteração.

```
Enquanto qntPares > 0 Faca
    Resto = (Resto * 100) + parAtual
    a = 0
```

Figura 19 CicloPrincipal

5. Ciclo Interno (Bloco Repetir...Até Falso):

- Este ciclo é o núcleo do cálculo. O seu objetivo é encontrar o maior dígito a (de 0 a 9) que funciona.
- Se (20 * Raiz + a) * a <= Resto Então: Esta é a condição de teste fundamental.

- Se for Verdadeiro: a = a + 1. O algoritmo incrementa a para testar o próximo dígito. O ciclo Repetir continua.
- Senao: Se a condição for Falsa (ou seja, (20 * Raiz + a) * a é maior que o Resto):
- a = a - 1: O a atual falhou, por isso o algoritmo recua para o valor anterior de a (que foi o último a funcionar).
- Interromper: O algoritmo sai do ciclo Repetir, pois encontrou o dígito a correto para esta etapa.

```
Repetir
    Se (20 * Raiz + a) * a <= Resto Entao
        a = a + 1
        Se a = 9 Entao
            Interromper
        Senao
            a = a + 1
        Fim Se
    Senao
        a = a - 1
        Interromper
    Fim Se
Ate Falso
```

Figura 20 Ciclo Interno

6. Atualização dos Valores:

- Assim que o Ciclo Interno é interrompido, o algoritmo tem o dígito a correto.
- Resto = Resto - (20 * Raiz + a) * a: O novo resto é calculado.
- Raiz = (Raiz * 10) + a: A Raiz é construída "anexando" o novo dígito a.

```
Resto = Resto - (20 * Raiz + a) * a
Raiz = (Raiz * 10) + a
```

Figura 21 Atualização dos Valores

7. Preparação para a Próxima Iteração:

- O algoritmo avança para o proximo par.
- O contador qntPares é diminuído em 1.
- O Ciclo Principal (Passo 4) recomeça.

```
Selecionar proximo par
```

Figura 22 Preparação para a Próxima Iteração

8.Resultado Final e Ajuste Decimal:

- O Fim Se fecha o bloco de lógica principal. Quando o ciclo Enquanto termina (todos os pares processados), o algoritmo faz o ajuste da vírgula.
- Inserir vírgula em Raiz depois de qntA dígitos: Esta ação corresponde diretamente ao passo "Colocar a virgula na Raiz após qntA digitos" do fluxograma.
- Escrever "Raiz ≈ ", Raiz e Escrever "Resto = ", Resto: Mostra os resultados.
- O Fim Se fecha o bloco de lógica e o programa termina

```
        Inserir virgula em Raiz depois de qntA digitos
        Escrever "Raiz ≈ ", Raiz
        Escrever "Resto = ", Resto
    Fim Se
Fim
```

Figura 23 Resultado Final e Ajuste Decimal

5. Exemplos práticos

1. Divisão Inteira

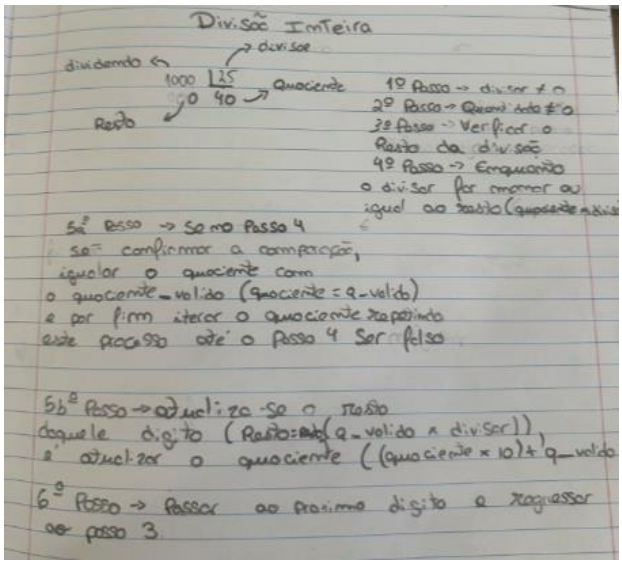


Figura 25 Exemplo

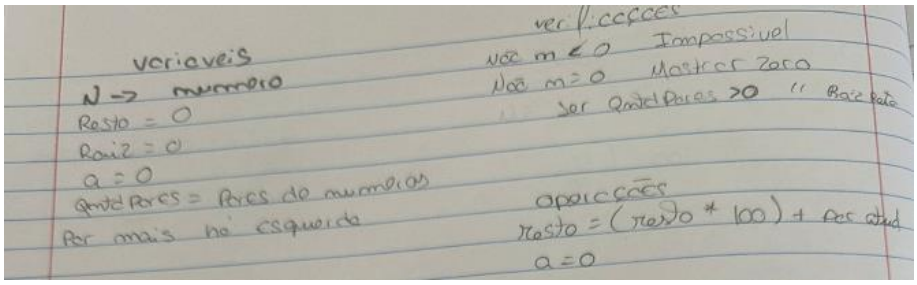


Figura 24 Exemplo

2. Raiz Quadrada

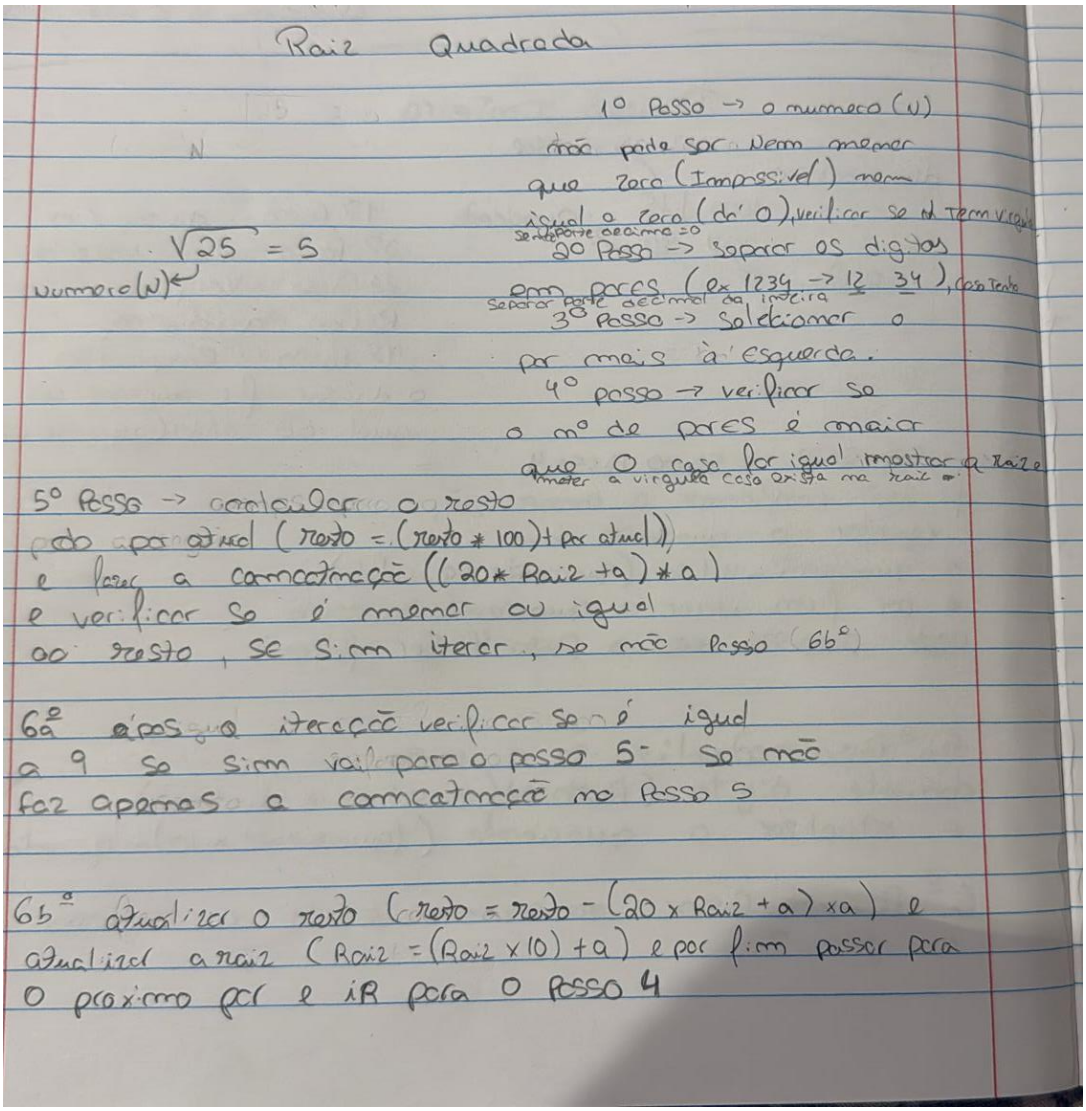


Figura 26 Exemplo

6.WEBGRAFIA

https://gemini.google.com
https://chatgpt.com/