PROFMAT-UNIVERSIDADE ESTADUAL DO SUDOESTE DA BAHIA

CINTHIA BATISTA NUNES

Introdução à computação: uma proposta para o ensino básico

Vitória da Conquista

2013

CINTHIA BATISTA NUNES

Introdução à computação: uma proposta para o ensino básico

Trabalho de Conclusão de Curso apresentado como requisito final para conclusão do Curso de Mestrado Profissional em Matemática – PROFMAT.

Área de Concentração: Matemática e suas tecnologias

Orientador: Prof. Dr. Sérgio Aguiar

Vitória da Conquista

2013

Nome: NUNES, Cinthia Batista
Título: Introdução à computação: uma proposta para o ensino básico

Trabalho de Conclusão de Curso apresentado como requisito final para conclusão do Curso de Mestrado Profissional em Matemática – PROFMAT.

Aprovado em:

Banca Examinadora

_____

Prof. Dr. Sérgio da Silva Aguiar
(Orientador - Presidente)
(UESB)

_____

Prof. Dr. Maria Deusa Ferreira da Silva
(Membro)
(UESB)

_____

Prof. Dr. Paulo Espinheira Menezes de Melo
(Membro)
(IFBA)

Dedicatória

Dedico este trabalho a meus familiares, especialmente meus pais, noivo, amigos e a meu professor Sérgio por todo o apoio para seguir nesta etapa.

Agradecimentos

Agradeço a Deus e a todos aqueles que compartilharam seus conhecimentos comigo.

Resumo

Este trabalho propõe o ensino de conceitos de linguagem de programação, como introdução à Ciência da Computação no ensino básico, numa abordagem simples e didática com vista ao uso de linguagens de alto nível tais como fluxogramas e pseudocódigos, como estímulo ao pensamento lógico matemático e computacional. Consideramos que o conhecimento acerca do funcionamento do computador é o ponto de partida para o aprofundamento desses conceitos. Acreditamos que o pensamento computacional deve ser desenvolvido nesses alunos, tendo em vista que o desenvolvimento das tecnologias depende da sua formação superior.

Palavras Chave: Algoritmos. Computação. Ensino. Fluxograma. Pseudocódigos.

Abstract

This paper proposes the teaching of programming language concepts, as introduction to computer science in basic education, in a simple and didactic approach you want to use high-level languages such as flowcharts and pseudocode, stimulate logical thinking and mathematical computing. We believe that knowledge about the functioning of the computer is the starting point for the further development of these concepts. We believe that computational thinking these students must be developed in order that the development of technologies depends on their higher education.

Keywords: Algorithms. Computing. Education. Flowchart. Pseudocodes.

LISTA DE TABELAS

LISTA DE FIGURAS

LISTA DE FLUXOGRAMAS

LISTA DE PSEUDOCÓDIGOS

Sumário

1 INTRODUÇÃO

A ideia de produzir um material que tenha como principal objetivo de abordar conceitos de computação no ensino básico surgiu de uma análise feita acerca desse momento de desenvolvimento de projetos e ações em prol do desenvolvimento das Ciências e Tecnologias. A sociedade tem exigido profissionais cada vez mais qualificados e capacitados a assumir uma postura autônoma, criativa, hábeis a solucionar problemas, com múltiplas inteligências e capazes de modificar aas ferramentas tecnológicas e o ambiente em que vivem.

São muitas as ações que o governo brasileiro tem executado para a melhoria da qualidade de ensino: programas de bolsas de estudo, implementação de novos cursos nas instituições de ensino técnico ou superior, criação de cursos técnicos, incentivos a qualificação, entre outros. E a cidade de Brumado na Bahia, tem ultimamente experimentado essas mudanças, com a criação de um do campos do Instituto Federal da Bahia (IFBA), a princípio, módulo de extensão do campos da cidade de Vitória da Conquista-BA.

Na cidade de Brumado, a instituição oferece cursos técnicos subseqüentes dentre eles o curso de informática com enfoque em programação web. Entretanto, a procura por este curso tem sido pequena e o índice de evasão dos alunos, ao contrário, tem se apresentado relativamente alto. E os estudantes têm responsabilizado suas dificuldades de aprendizado em disciplinas com Matemática e Linguagem de Programação pela evasão. Contudo, apesar das ações de estimulo do governo, muitos estudantes têm chegado ao ensino técnico e superior sem ter desenvolvido o raciocínio lógico e a capacidade de resolver problemas.

Por vivenciar esta realidade e entender que o progresso das ciências e tecnologias depende também da formação de profissionais de ciências da computação e da qualidade dessa formação, é que propomos uma introdução à computação no ensino básico. Por meio da abordagem de conceitos que podem ser trabalhados desde o ensino fundamental até o ensino médio e que acreditamos estimular o raciocínio lógico e contribuir com o desenvolvimento de habilidades ligadas à resolução de problemas. Além, de proporcionar acesso ao pensamento algorítmico e computacional.

Tratamos do ensino-aprendizado de algoritmos por meio da construção e análise de fluxogramas e pseudocódigos. Estaremos assim, viabilizando não só o aprendizado da Computação básica e de programação, como também da própria Matemática. Iniciamos nosso trabalho com uma breve descrição do funcionamento do computador, por meio de uma analogia proposta por Terada e Setzer (1992), chamado de Computador à Gaveta,

apresentando as características básicas de um computador de forma clara e acessível a nível básico, como introdução ao estímulo do pensamento computacional.

As ferramentas que utilizaremos para atingir nossos fins são os fluxogramas e pseudocódigos *(Portugol)*, consideradas linguagens de alto nível, por serem de fácil entendimento e possuírem uma estrutura próxima das linguagens usualmente utilizadas por programadores, o que aproxima os estudantes destes profissionais.

Este procedimento pode ser feito durante as aulas de Matemática, ao tempo que o professor planeje iniciar a abordagem de novos conceitos ou revisar conhecimentos prévios. Senão, por meio da inserção de uma nova disciplina que contemple conteúdos da computação básica com foco na introdução à linguagem de programação, desde o ensino fundamental, até o ensino médio, visando tanto minimizar e sanar dificuldades apresentadas por futuros alunos de matemática e computação quanto contribuir para o progresso das Ciências Tecnológicas.

2 REFERENCIAL TEÓRICO

Hoje em dia, muito se fala em aplicação das Tecnologias da Informação e Comunicação - TIC's, no ensino e aprendizado de estudantes do nível médio e fundamental. O uso de ferramentas como o computador, internet, vídeo e outros recursos audiovisuais tornou-se fundamental para a aquisição do conhecimento através do desenvolvimento de atividades educacionais. A maioria dos educadores possui uma grande preocupação em integrar na construção de suas atividades, procedimentos e conteúdos que sejam verdadeiramente úteis e valorosos, para o progresso das Ciências e da sociedade:

> Não se pode negar o impacto provocado pela tecnologia da informação e comunicação na configuração da sociedade atual. Por um lado, tem-se a inserção dessa tecnologia no dia-a-dia da sociedade, a exigir indivíduos com capacitação para bem usá-la; por outro lado, tem-se nessa mesma tecnologia um recurso que pode subsidiar o processo de aprendizagem da Matemática. É importante contemplar uma formação escolar nesses dois sentidos, ou seja, a Matemática como ferramenta para entender a tecnologia, e a tecnologia como ferramenta para entender a Matemática. (SECRETARIA DE EDUCAÇÃO BÁSICA, 2008, p. 87)

Não raro, para estimular e fazer acontecer o ensino-aprendizado de matemática e áreas afins, muitos pesquisadores vêem criando e testando estratégias aplicáveis em sala de aula com o propósito de sanar dificuldades de aprendizado. Consta em muitas dessas pesquisas, que muitos estudantes têm desenvolvido a capacidade de reproduzir passos para atingir objetivos, como por exemplo, para solucionar problemas, mas são em maioria, incapazes de criar estes passos e mesmo aqueles que conseguem construir com autenticidade a solução de um problema, tem dificuldade de expor ou demonstrar com clareza o seu processo de raciocínio.

Siebra e Silva (2009, p. 87) reafirmam nosso pensamento, quando destacam que:

> "[...] podemos observar que muitos alunos são capazes de reproduzir algoritmos prontos, mas são incapazes de efetuar modificações para adequá-las a pequenas alterações nas condições do problema e, também, não conseguem minimamente começar o desenvolvimento de um algoritmo para resolver problemas considerados fáceis do ponto de vista das estruturas lógicas envolvidas. Essas observações são indícios de dificuldades com a resolução de problemas em si, com o entendimento do problema e o delineamento dos passos para resolvê-lo [...]".

Nesta perspectiva, de preocupação com o ensino e aprendizado de Matemática, é que surgiu a proposta de um trabalho que integre elementos da computação no Ensino Básico.

Não estamos falando aqui de aprender manusear computadores ou outras tecnologias, porque acreditamos que boa parte da sociedade educativa tecnológica já tenha se preocupado em renovar sua prática pedagógica utilizando novas ferramentas. Nosso interesse é relacionar, em nível elementar, o ensino de Matemática com noções de computação básica, tendo em vista dois aspectos: (1) boa parte dos estudantes do Ensino Básico e técnico acredita que a Matemática e a Ciência da Computação nada têm haver uma com a outra; (2) a compreensão do funcionamento desses recursos, bem como o seu modo de processar as informações. Considerando que um computador executa tarefas definidas pelo programador, por meio de instruções pré-estabelecidas numa linguagem de programação e esse conhecimento pode contribuir para o desenvolvimento do raciocínio lógico, da criticidade e autonomia, possibilitando a esses alunos o aprendizado de noções básicas de computação.

Dito isso, propomos neste trabalho a introdução da linguagem de programação no Ensino Básico, com foco na construção, análise e aplicação de algoritmos, por entendermos que o desenvolvimento das ciências tecnológicas depende da afinidade e habilidade que estes jovens adquirem em compreender processos de construção e execução de recursos computacionais. Tendo em vista que:

> No estudo de algoritmos, conceitos como abstração/refinamento, modularização, recursão/iteração, etc., podem ser aplicados às outras ciências, ampliando a capacidade de raciocínio na resolução de problemas, por meio de processos de aprendizagem metacognitivos, considerados aspectos chaves da inteligência. (NUNES, 2011, p. 1)

Ou seja, pretendemos estimular nestes estudantes, o *pensamento computacional*. Blikstein (2008, p. 1) define *pensamento computacional* como "[...] o saber usar o computador como um instrumento de aumento do poder cognitivo e operacional humano, isto significa usar computadores e redes de computadores para aumentar a produtividade, inventividade e a criatividade".

A implementação do *pensamento computacional* no ensino básico é incentivada pela Sociedade Brasileira de Computação (SBC) há muitos anos. A SBC salienta a importância da qualidade do ensino em todos os níveis e aborda que a Computação deve ser ensinada desde o ensino fundamental, assim como as outras Ciências, para que tenhamos recursos humanos qualificados para enfrentar os desafios que estão por vir.

Existe uma carência muito significativa de estudantes interessados nessa área e entre aqueles dispostos a seguir carreira neste campo permanece uma maioria que apresenta dificuldades de aprendizado. Segundo Nunes (2011, pag 2), "[...] a demanda por profissionais

de computação cresce exponencialmente a cada ano e as universidades não estão formando profissionais na mesma proporção". Dessa maneira faz-se necessário aumentar o interesse por cursos de computação, o que requer ações de estímulo pela introdução da computação na educação básica.

A Lógica de Programação é uma disciplina fundamental na grade curricular do curso de Ciência da Computação, seja ele num nível mais básico (técnico) ou avançado (superior), e exige dos alunos, uma postura de caráter investigativo com habilidade em solucionar problemas.

Siebra e Silva (2009, p. 25) ressaltam a importância da Lógica de Programação e afirmam:

> A importância dessa disciplina é percebida tão logo o aluno inicie o aprendizado de linguagens de programação e precise ordenar os passos para resolução de um problema, repassando-os ao computador, sob a forma de comandos ou instruções de linguagens de programação específicas. Isso porque a base para a programação, independente da linguagem utilizada e da finalidade, é a construção de algoritmos e o raciocínio lógico empregado na construção dos mesmos [...].

Ainda de acordo com Siebra e Silva (2009, p. 8), "[...] construir algoritmos requer que o aluno consiga usar suas habilidades e competências aplicadas à resolução de problemas da vida cotidiana na resolução de problemas computacionais".

Com foco na resolução de problemas, educadores matemáticos ao longo dos séculos, apresentaram diferentes concepções no modo de fixar o objetivo de se aprender e ensinar sobre Matemática e resolução de problemas no ensino básico, transitando da ideia de ensinar *sobre* resolução de problemas (iniciada em torno de 1980), até o ensinar *por meio* da resolução de problemas (concepção dos dias atuais), passando pelo ensinar *para* resolver problemas (ONUCHIC; ALLEVATO 2011, p. 79).

Embora estas concepções tenham enfoque em desenvolver metodologias sobre o tratamento da resolução de problemas em sala de aula, cuja preocupação reside no ensino-aprendizado de matemática, nota-se a escassez de uma estratégia, que acreditamos contribuir para o desenvolvimento do raciocínio lógico, valorizando a forma de pensamento dos alunos: a (re) produção de forma mais clara, objetiva, finita, dos passos lógicos utilizados para o alcance da solução do problema, ou seja, a construção e análise dos algoritmos envolvidos.

Ao inserir o estudo de algoritmos no Ensino Médio, estamos fornecendo a essa nova geração acesso ao pensamento algorítmico e computacional como um avanço do pensamento lógico matemático. Os alunos serão induzidos a criar, planejar, analisar, e executar ações

aplicando conhecimentos prévios e buscando novos conhecimentos para responder às suas questões.

Siebra e Silva (2009, p.27), apontam que:

> A resolução de problemas computacionais envolve várias capacidades e habilidades: abstração, modelagem, criatividade, organização, seqüenciamento de idéias e com capacidade de descrever sem ambigüidades como realizar alguma ação. Tornar-se apto na resolução de problemas computacionais exige um percurso complexo que se inicia na compreensão da proposta problema, passa pela descrição de um algoritmo, utilizando-se uma linguagem de pseudocódigo ou diretamente em uma linguagem de programação e finaliza com testes de corretude da solução desenvolvida [...].

Não precisamos esperar que um estudante alcance a graduação para adquirir tais competências que podem e devem ser desenvolvidas desde o Ensino Básico. Pois, torná-lo apto na resolução de problemas não é apenas o objetivo do ensino da matemática, mas de todas as outras disciplinas. E aprender construir algoritmos, requer aprender o que há de mais essencial na solução de um problema, pois o algoritmo detém em sua estrutura o como e o porquê é possível ou não atingir o resultado desejado.

3 INICIAÇÃO BÁSICA AO PENSAMENTO ALGORITMO E COMPUTACIONAL

3.1 Modelo Simples de um Computador

A partir da década de quarenta começaram a surgir máquinas que executavam, mesmo que precariamente no início, operações matemáticas previamente programadas. Ou seja, executavam processos bem definidos matematicamente até obter-se um resultado; Estas máquinas receberam o nome de computador.

Para que possamos ‘fazer funcionar’ uma máquina ou equipamento, devemos conhecer o seu funcionamento; devemos conhecer o que ela pode executar e de que forma ela executa tarefas.

Para conhecermos basicamente como funciona um computador vamos empregar um **modelo imaginário**, chamado **computador à gaveta**, introduzido por Terada e Setzer (1992). Este modelo faz uma analogia de suas funções com as funções de um computador real (eletrônico), apresentando as características básicas deste último, porém com um mecanismo suficientemente simples para ser compreendido.

3.1.1 O Computador à Gaveta

O computador à gaveta é formado por 6 componentes:

1) *Um gaveteiro com 99 gavetas:* consiste numa sequência de 99 gavetas numeradas de 1 a 99; o número de cada gaveta é denominado seu **endereço**. Cada gaveta contém um quadro negro, onde é escrito um número sempre com 3 algarismos. As regras de utilização do **gaveteiro** são as seguintes:

a) Em qualquer momento, no máximo uma gaveta pode estar aberta;

b) A leitura do quadro-negro de uma gaveta não altera o que nele está gravado;

c) A escrita de uma informação no quadro-negro em uma gaveta é sempre precedida do apagamento do mesmo;

d) Somente o **processador** tem acesso ao gaveteiro.

O computador eletrônico moderno possui, em vez de um gaveteiro, um dispositivo eletrônico denominado *memória*, com regras de funcionamento análogas às do gaveteiro. Este dispositivo contém partes, denominadas palavras, que correspondem às gavetas do gaveteiro.

2) *Uma calculadora*: é uma calculadora comum,com teclado para entrada de números, teclas para operações aritméticas básicas, tecla '=' e um mostrador, que denominaremos de **acumulador**. Há dois tipos de operações efetuadas com esta calculadora:

a) Carga de um número no acumulador. Para isso, pressiona-se a tecla '=' (garantindo o encerramento de alguma operação prévia) e a seguir 'digitam-se' os algarismos do número a ser carregado, o qual aparece no acumulador;

b) *Operações aritméticas*. Esta é sempre feita entre o número que está no acumulador e um segundo número. A calculadora só pode ser usada pelo **processador**.

No computador moderno, a calculadora eletrônica chamada de *unidade aritmética*. Em lugar de teclas, as operações aritméticas são acionadas eletronicamente. Por exemplo, ao ser executada uma instrução de soma, um impulso eletrônico é encaminhado pela unidade de controle à seção apropriada do circuito da unidade aritmética, iniciando uma sequência de operações eletrônicas que resultam na obtenção do resultado no acumulador. O acumulador é um 'registrador' acessível eletronicamente à unidade de controle, isto é, não tem exibição visual.

3) *Um quadro-negro denominado EPI*: independente do gaveteiro, com a forma □□, onde será escrito um número entre 00 e 99, correspondente a um endereço da gaveta do gaveteiro. O número nele escrito indica o 'Endereço da Próxima Instrução'. Somente o **processador** tem acesso ao EPI.

Nos computadores reais de hoje o Endereço da Próxima Instrução é um registrador eletrônico, sem exibição visual. Somente a unidade de controle tem acesso ao EPI, podendo consultá-lo ou alterar seu conteúdo.

4) *Porta-cartões*: é um dispositivo onde ficam empilhados cartões. O porta-cartões funciona do seguinte modo:

a) Cartões com informação são colocados exclusivamente na parte superior, um a um. Números são escritos sempre com 3 algarismos;

b) Cartões são retirados da extremidade inferior (da pilha) um de cada vez, sempre pelo **processador**

c) A colocação de cartões só pode ser feita pelo USUÁRIO (pessoa que utiliza o computador)

Em lugar do 'Porta Cartões' o computador moderno possui uma unidade de entrada com capacidade para ler eletronicamente *linhas de entrada* com o formato adequado.

Vários dispositivos podem ser usados como unidade de entrada: cartões perfurados (onde os furos codificam os elementos da linha), cartões marcados a lápis, teclado como o de máquina de escrever, etc. Todos eles transformam a representação externa, acessível ao usuário, em impulsos eletrônicos que são enviados pela unidade de controle à memória, posicionando os circuitos dessa para que as palavras envolvidas tenham um conteúdo equivalente à representação externa.

5) *Folha de Saída:* folha de papel onde pode ser escrito um número em cada linha, utilizando-se sempre linhas consecutivas. Somente o processador pode escrever nessa folha. Somente o usuário lê o que já foi escrito.

Em lugar da 'Folha de Saída' o computador moderno possui uma unidade de saída com capacidade de gravar linhas de saída. O dispositivo de saída pode ser uma máquina de escrever, uma impressora por linha (isto é, que imprime as **n** posições da linha simultaneamente), um terminal de vídeo, etc.

6) *Operador*: executa estritamente ordens recebidas, não podendo tomar nenhuma iniciativa própria. É quem denominamos **processador**.

O **processador** trabalha sempre em um dos dois estados diferentes:

a) *Estado de carga*: onde ele exclusivamente transcreve informações de cartões, lidos no porta-cartões, para gavetas no gaveteiro.

b) *Estado de Execução*: onde ele executa ordens gravadas nas gavetas.

A comunicação entre usuário e operador é feita exclusivamente através das unidades, porta-cartões e folha de saída.

O **processador** sabe fazer 'de cabeça' uma única operação aritmética: incrementar de 1 o conteúdo do EPI.

Nos computadores eletrônicos (reais), o papel de acionar outros dispositivos (gaveteiro, calculadora, EPI, etc) é desempenhado por um sistema de circuitos eletrônicos,

cujo funcionamento equivale a ações executadas pelo **processador** do gaveteiro. Esse sistema é denominado de *Unidade de Controle* (em inglês, "Central Processing Unit" ou CPU).

3.1.2 O Conceito de Programa

Para resolver um problema usando o computador a gaveta, o usuário deve planejar uma sequência de ordens a serem executadas pelo **processador**. Cada uma dessas ordens é denominada instrução. Um exemplo de instrução é o seguinte: "some o conteúdo da gaveta do endereço 41 ao conteúdo do acumulador". A fim de produzir-se a execução correta das instruções e na sequência adequada, elas são escritas na gaveta do GAVETEIRO. Para executar uma instrução da sequência, o **processador** segue os seguintes passos:

a) Consulta o EPI, onde está escrito o endereço E da próxima instrução;
b) Incrementa de 1 o conteúdo do EPI, apagando o valor anterior e escrevendo o novo valor (o qual neste caso será E + 1);
c) Abre a gaveta de endereço E, nesta gaveta ele deve encontrar uma instrução I, que é lida no quadro-negro da gaveta.
d) Fecha a gaveta E;
e) Executa I.

Após finalizados esses passos, o **processador** recomeça do passo (a), com a exceção de casos excepcionais a serem explicados adiantes. Se a execução da instrução I não acarretar alteração no conteúdo do EPI, a próxima instrução a ser executada será a da gaveta de endereço E + 1, devido ao passo (b). Se uma instrução acarretar alteração EPI, mudando o seu conteúdo para X, a próxima instrução a ser executada será a da gaveta de endereço X; diz-se que houve um desvio para X. Com isso, as instruções a serem executadas pelo **processador** não estão necessariamente em gavetas de endereço consecutivos.

Em particular, a execução da instrução da gaveta 10, que poderia ser "altere o EPI para 5" ou equivalente, "desvia para 5", produziria um desvio "para trás", o que possibilita a repetição da execução das instruções que estão nas gavetas de endereço 5 e 10. Assim, uma subsequência de instruções pode ser executada um número *n* qualquer de vezes, sem necessidade de se ter *n* cópias dessa subsequência no gaveteiro.

As instruções escritas nas gavetas do GAVETEIRO constituem um *programa armazenado*. O usuário deve formular o método de resolução de seu problema sob a forma de um *programa*, que é, portanto, um conjunto (finito) de instruções. Para conseguir a execução de um programa, o usuário deve produzir inicialmente o armazenamento desse programa no gaveteiro, passando, portanto, a constituir um *programa armazenado*. Isso é feito da seguinte forma:

a) O usuário escreve cada instrução em um cartão, precedida de um endereço; assim, cada cartão do programa contem um par ordenado (E,I), onde E é um endereço e I uma instrução;
b) O **processador** é colocado em *estado de carga de programa*;
c) O usuário coloca um conjunto de cartões do programa no porta-cartões, em qualquer ordem;
d) Como o **processador** está em estado de carga, ele lê um cartão com um par (E,I); abre a gaveta de endereço E; escreve em seu quadro-negro a instrução I; fecha a gaveta;
e) O **processador** repete o passo (d) até ler o ultimo cartão de programa, após a leitura ele é colocado em *estado de execução de programa*.

Após o encerramento da carga do programa, o **processador** é colocado em *estado de execução de programa*. Isso é feito por meio de um cartão especial, que deve encerrar o conjunto de cartões de programa. A forma desse cartão é //EXECUTE *X*, onde *X* é um número escrito pelo usuário; será o endereço da gaveta onde se encontra a primeira instrução a ser executada.

Ao ler esse cartão, o **processador** apaga o EPI e escreve no mesmo valor *X*; a seguir, ele vai para o passo (a) da execução de uma instrução, como exposto no inicio deste item.

Para completar este quadro, resta descrever como o **processador** entra em *estado de carga de programa*. Vamos supor que na verdade esse estado seja o estado "normal" do **processador**; ele só pode sair desse estado ao tentar carregar um cartão //EXECUTE. Estando em estado de execução, ele só sai desse estado nos dois casos seguintes:

a) Através da execução da instrução "pare a execução";
b) Se ocorrer algum erro durante a execução.

Um exemplo de caso (b) é o de o **processador** tentar executar uma instrução inválida, isto é, não conhecida.

3.1.3 Exemplos de Instruções

Neste item apresentamos esquematicamente os passos de execução de algumas instruções que serão empregadas no exemplo de um programa a ser apresentado na subseção 3.1.3.1.

O conteúdo de uma gaveta de endereço E, isto é, número gravado em seu quadro-negro, será representada por cE. Assim, c10 indicará o conteúdo da gaveta 10. Indicaremos por cAC o conteúdo do acumulador; este será abreviado por AC.

i. Soma

- Instrução: "some o cE ao AC".
- Significado: some o cE ao cAC e coloque o resultado no AC; o cE não se altera.
- Execução: o **processador** efetua os seguintes passos:

a) digita a tecla '+' da CALCULADORA;
b) abre a gaveta de endereço E;
c) lê o número escrito nessa gaveta (cE) e digita-o na CALCULADORA;
d) fecha a gaveta E;
e) digita '=' na CALCULADORA.

Daqui em diante, em lugar de escrevermos "gaveta de endereço E", escreveremos simplesmente E. Também deixaremos de mencionar explicitamente que é o **processador** quem efetua os passos de execução.

ii. Carga no AC

- Introdução: "carregue o cE no AC".
- Significado: copie o cE no AC; o cE não muda.
- Execução:

a) digita '=';
b) abre E;

c) lê cE e digita-o;

d) feche E.

iii. Armazenamento do AC

- Instrução: "armazena o cAC em E".
- Significado: copie o cAC em E; o cAC não muda (oposto da instrução anterior).
- Execução:

a) abre E;

b) apaga o cE;

c) lê o cAC e o escreve em E;

d) feche a gaveta.

iv. Impressão

- Instrução: "imprima o cE".
- Significado: o cE é transcrito na FOLHA DE SAÍDA.
- Execução:

a) abre E;

b) lê cE e escreve o seu valor na próxima linha da FOLHA DE SAÍDA;

c) fecha E.

v. Leitura

- Instrução: "leia um cartão e guarde em E".
- Significado: o conteúdo do próximo cartão do PORTA-CARTÕES é lido e transcrito para E.
- Execução:

a) abre E;

b) retira um cartão do PORTA-CARTÕES;

c) lê o conteúdo do cartão e escreve o seu valor em E;

d) joga fora o cartão;

e) fecha E.

Note que supusemos haver cartão no PORTA-CARTÕES. Em caso contrário, o **processador** aguarda a colocação de pelo menos um cartão no PORTA-CARTÕES.

vi. Desvio condicional

- Instruções: “se cAC ≠ 0, desvie para E”.
- Significado: se há um número diferente de 0 no AC, a próxima instrução a ser executada esta em E, caso contrário não há nada a fazer (isto é, a próxima instrução a ser executada estará na gaveta seguinte à que contém esta instrução).
- Execução:

a) lê o cAC;

b) se cAC ≠ 0 então apaga o EPI e escreve E no mesmo.

vii. Pare

- Instrução: “pare”.
- Significado: encerra a execução do programa.
- Execução:

a) entrega a FOLHA DE SAÍDA para o usuário;

b) entra no estado de carga.

3.1.3.1 Exemplo de Programa

Seja o seguinte problema:

É dada uma sequência de números inteiros positivos, determine sua soma.

Vamos supor que o usuário do computador gaveta planeje resolver este problema da seguinte maneira (uma explanação do porquê desse esquema é dada mais adiante):

a) Cada número da sequência é escrito em um cartão;

b) Dois cartões adicionais contendo o número 0 são colocados um imediatamente antes do primeiro cartão da sequência, e o outro logo após o último cartão;

c) O programa abaixo é escrito em cartões já no formato de carga de programa da tabela 1;

d) É formada uma pilha de cartões com a seguinte ordem: programa //EXECUTE 01 - cartões conforme (a) e (b) acima. Essa pilha é colocada no PORTA-CARTÕES. Teremos nesta unidade, portanto, os cartões denominados cartões de programa e cartões de dados, precedendo e seguindo, respectivamente, o cartão //EXECUTE;

Tabela 1 – Programa da soma de uma sequência de inteiros positivos

Endereços	Instrução
01	leia um cartão e guarde em 11
02	leia um cartão e guarde em 12
03	imprima o c12
04	carregue no AC o c11
05	some o c12 ao AC
06	armazene o cAC em 11
07	carregue o c12 no AC
08	se cAC $\neq 0$, desvie para 02
09	imprima o c11
10	Pare

Fonte**: Terada e Setzer (1992)**

e) Terminada a execução, é recebida a FOLHA DE SAÍDA, onde estarão impressos os números da sequência, seguidos da soma procurada.

3.1.3.2 Esquemas Estruturados

Esquemas estruturados são constituídos de ações denominadas *comandos*. Os comandos, por sua vez, representam processos a serem executados internamente pela máquina. As instruções de máquina do programa anterior podem ser transformadas no esquema estruturado da tabela 2:

Tabela 2 – Esquema estruturado

Endereço	**Instrução**	**Esquema estruturado**
01	leia e guarde em 11	SOMA : = 0
02	leia e guarde em 12	Repita
		Leia número
03	grave o c12	Escreva NÚMERO

04	carregue no Ac o c11	SOMA : = SOMA + cNÚMERO
05	some o c12 ao AC	
06	armazene cAC em 11	Até que cNÚMERO =0
07	carregue no AC o c12	
08	se cAC ≠ 0 desvie para 02	
09	grave o c11	Escreva cSOMA
10	pare	Fim

Fonte: Terada e Setzer (1992)

Um esquema estruturado pode consistir de um fluxograma ou pseudocódigo, e apresenta comandos de *atribuição*, *entrada*, *saída*, *repetição* e *seleção simples* que serão definidos no capítulo seguinte.

3.2 Algoritmos

Estamos habituados a executar, diariamente, passos lógicos, que nos possibilita solucionar problemas do cotidiano, ou simplesmente atingir objetivos específicos. São ações que necessitam ser executadas em devida ordem, para que não resulte num desastre. Atividades como cozinhar, se preparar para o trabalho, consertar ou montar um móvel ou eletrodoméstico, são exercícios básicos, que estamos sujeitos a fazer diariamente e cuja omissão de determinados processos, pode acarretar em um inconveniente. "Um **algoritmo** é uma sequência de passos necessária para resolver qualquer problema." (FARREL, 2010, p. 399).

Um algoritmo deve ser eficiente, todas as operações devem precisamente definidas e suficientemente básicas de modo que possam ser em princípio, executadas com precisão em um tempo finito.

Encontra-se no site Wikipédia, as seguintes considerações sobre algoritmos:

Geralmente assume-se que um algoritmo deve satisfazer os seguintes requisitos:

- O algoritmo consiste de um conjunto finito de instruções simples e precisas, que são descritas com um número finito de símbolos.
- O algoritmo sempre produz resultado em um número finito de passos.

- O algoritmo pode, a princípio, ser executado por um ser humano com apenas papel e lápis.
- A execução do algoritmo não requer inteligência do ser humano além do necessário para entender e executar as instruções.

De acordo com Brookshear (2003, p 151):

> [...] os passos em um algoritmo devem possuir uma estrutura bem estabelecida em termos da ordem na qual são executadas. Isso não implica que os passos sejam executados em sequência consistam no primeiro passo, seguido do segundo e assim por diante. Alguns algoritmos, conhecidos como algoritmos paralelos, por exemplo, apresentam mais de uma sucessão de passos possível, cada qual projetada para ser executada por processadores diferentes em uma máquina com multiprocessamento.[...].

Siebra e Silva (2009, p. 14) consideram as seguintes fases de construção de um algoritmo:

- **Entrada:** são os dados de entrada do algoritmo. As informações que ele vai precisar para poder solucionar o problema.
- **Processamento:** são os procedimentos utilizados para chegar ao resultado final, tais como cálculos, conversões, operações, etc.
- **Saída:** são os dados já processados que, geralmente, serão apresentados aos usuários.

Um algoritmo pode ser constituído por três tipos de estruturas: seqüencial, condicional e repetição.

- Estrutura seqüencial (incondicional): corresponde a um conjunto indeterminado, porém limitado, de ações que devem ser executadas, todas passo a passo, uma após a outra.
- Estrutura condicional ou estrutura de seleção: com essa estrutura, você faz uma questão, e, dependendo da resposta, toma um de dois modos de ação.
- Estrutura de repetição ou loop: corresponde a um conjunto de ações que deverá ser repetido inúmeras vezes até que a condição seja falsa, daí, o fluxo de execução dará continuidade ao restante das ações.

Os fluxogramas e pseudocódigos são ferramentas capazes de representar quaisquer processos, assim como os algoritmos e até mesmo os não matemáticos, permitindo-nos uma boa aproximação da linguagem computacional e tornado-se bons recursos para a compreensão da solução de problemas, de forma a contribuir para a elaboração do pensamento matemático.

3.2.1 Fluxogramas

Conforme Farrel (2010, p. 14), "**diagramas de blocos** ou **fluxogramas** são representações gráficas de passos lógicos (algoritmos) a serem tomados para resolver um problema". Os fluxogramas exibem claramente a possível sequência de passos, ilustrando visualmente o processo. Ao ente que executa os passos de um processo qualquer, chamaremos de **processador.** Assim, um processador, poderá ser uma pessoa ou uma máquina.

Mesmo num nível de primeiro grau, os fluxogramas constituem para alunos mais jovens uma introdução adequada à análise de problemas, métodos operacionais e programação. Para usar esta ferramenta com sucesso, necessitamos inicialmente, suscitar em nossos alunos uma postura de verdadeiros resolvedores de problemas, pois é preciso planejar com antecedência e antecipar dificuldades quando formos executar uma atividade que constitua ou não problema para nós.

Fluxogramas apresentam uma estrutura básica:

1. Início e fim que serão aqui representados por uma forma oval:

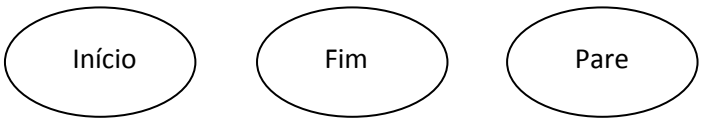


Figura 1 – Estrutura básica inicio e fim

2. Bloco de ações ou operações nos quais anotamos o que está por fazer, aqui representados por retângulos:

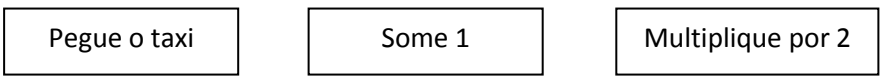


Figura 2 – Estrutura básica do bloco de ações

3. Bloco de decisões usado para denotar as decisões que precisamos tomar antes de prosseguir.

Utilizaremos o bloco de decisões sempre que estivermos diante de questões que apresentem alternativas como, por exemplo, "se a resposta for *sim* então faça isso, se a resposta for *não* então, faça outra coisa." Estes blocos constituem de uma parte importante do diagrama e serão representados por um losango:

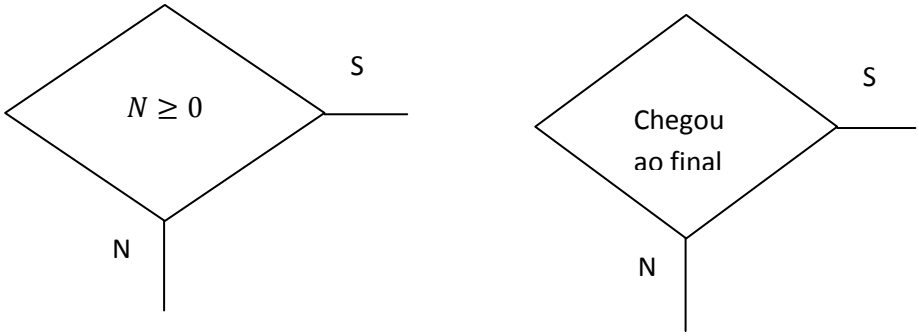
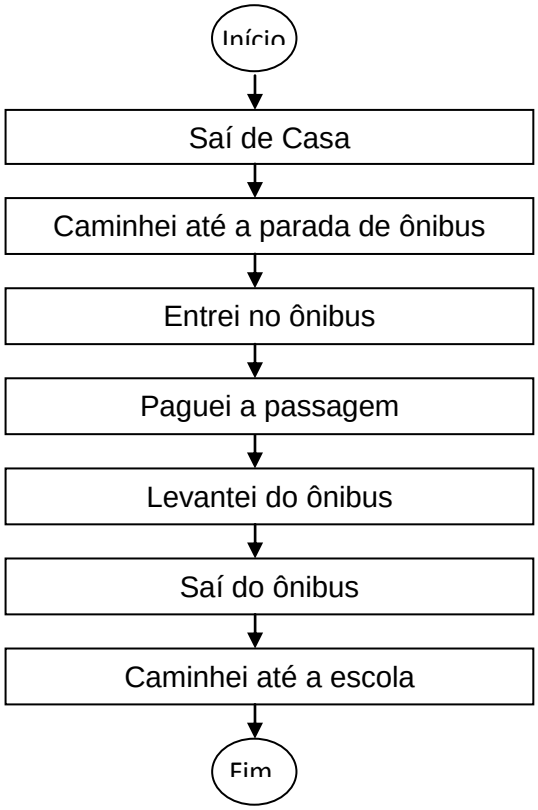


Figura 3 – Estrutura básica do bloco de decisão

Para conectar as sentenças acima descritas, de forma ordenada, utilizaremos aqui setas ou linhas de fluxo. Além disso, "[...] a maior parte do fluxograma deve ser legível de cima para baixo ou da esquerda para a direita em uma página. Essa é a nossa forma de leitura, então quando fluxogramas seguem essa convenção, eles são mais fáceis de entender." (FARREL, 2010, p. 16)

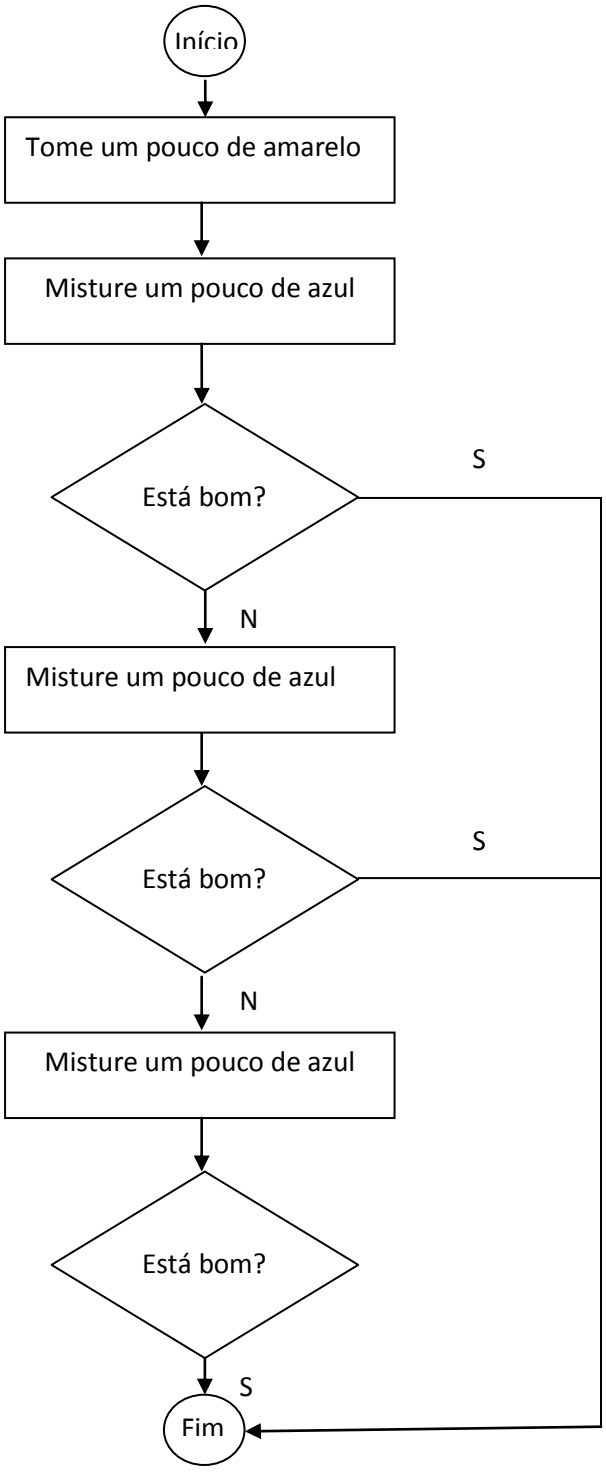
Atividades bem simples que podem ser descritas num fluxograma e aplicadas em sala de aula para introduzirmos o ensino de algoritmos.

Exemplo 1: Podemos solicitar que os alunos descrevam os passos ou ações mais importantes desde a sua saída de casa até chegar à escola, de forma ordenada.



Fluxograma 1 –Rotina para chegar a escola

Exemplo 2: A preparação de uma tinta (num certo tom de verde) pode ser descrita da seguinte maneira:



Fluxograma 2 – Preparação de tinta

Observe, que no fluxograma 2, construído para preparar uma tinta, executamos a ação "misture um pouco de azul" mais de uma vez (até atingirmos o resultado esperado. Em programação, esta ação, executar os passos repetidas vezes, até chegar ao resultado esperado é

chamado de ‘laço’ *(loop)* e pode ser ilustrada no fluxograma por uma estrutura de controle que substituirá a parte repetitiva. Esta estrutura, poderá ser colocada no início ou no final de um fluxograma.

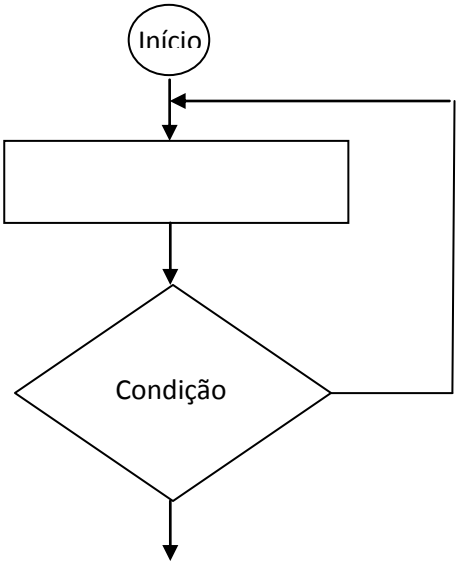


Figura 4 – Controle de início

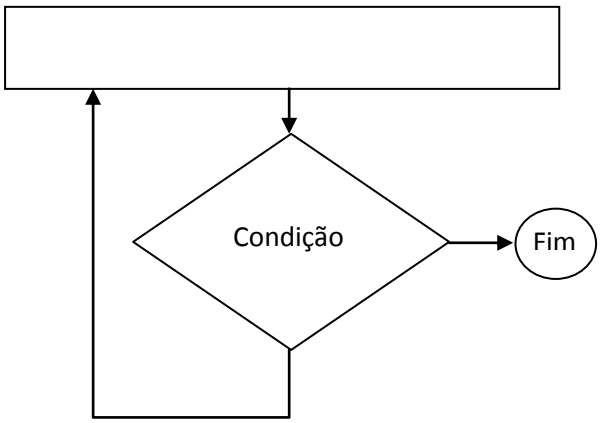
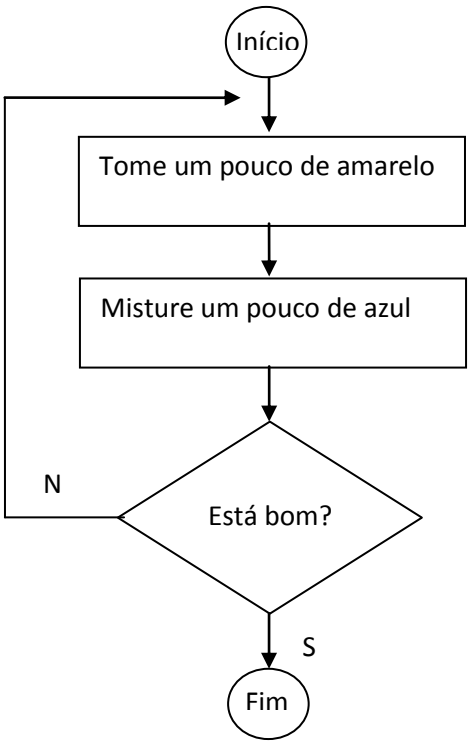


Figura 5 – Controle no final

Assim, podemos substituir o fluxograma 2 (preparação da tinta) pelo fluxograma 3:



Fluxograma 3 – Preparação de tinta com repetição

Num nível mais elementar, inicialmente, não esperamos que crianças consigam construir fluxogramas com precisão. Entretanto, apresentá-las fluxogramas para interpretar acarretará numa experiência valiosa. Esta abordagem pode ser utilizada como uma alternativa para introduzir determinado assunto, além disso, quaisquer atividades que possam ser organizadas sob a forma de diagrama, como por exemplo, resolução de quebra-cabeças, confecção de animais e objetos de papel, e roteiros para desenhar figuras geométricas podem ser descritas sob a forma de fluxogramas. Cabe ao professor, sugerir aos alunos, trabalhos de casa envolvendo a descoberta de situações diárias que possam ser organizadas utilizando esta ferramenta.

3.2.2 Pseudocódigos

Assim como os fluxogramas, os pseudocódigos são representações lingüísticas, de passos lógicos a serem executados para resolver um problema. Constitui-se claramente de uma receita, escrita na linguagem natural ou simbólica, mas que possui uma estrutura bem próxima

das sentenças escritas em linguagem de programação. Quando fazemos o uso excessivo da língua portuguesa para representar o pseudocódigo, este recebe o nome de **Portugol.**

Em geral, inicia-se um pseudocódigo com uma declaração tipo "Início" e termina com uma declaração tipo "Fim". Entre o início e o fim, um pouco deslocados, são descritos os passos para resolução do problema ou o algoritmo que se deseja executar, de modo que "Início" e "Fim" se sobressaiam. Em Ciências da Computação a ação de deslocar sucessivamente os passos de um algoritmo é chamada de **indentação**. O pseudocódigo 1 indica se o dia em questão refere-se à data de aniversário de uma determinada pessoa:

Início
    Leia dia
    Se dia for igual a 04/10 então
Indentação *escreva: Hoje é meu aniversário!*
    Senão
Indentação *escreva: Hoje não é meu aniversário.*
    Fim do se
Fim

Pseudocódigo 1 – Data de nascimento

Note que para representar a estrutura de seleção no pseudocódigo anterior usamos as expressões "se...então", "senão".

O pseudocódigo 2 descreve uma possível ação, que um estudante pode executar diariamente, como fizemos para exemplificar os fluxogramas. É interessante que os estudantes percebam que a omissão de alguns passos acarretará na execução errônea daquela ação e que para uma máquina como o computador, as ações devem estar claramente explicitadas, caso contrário, comprometeria o resultado final:

Início
    Acorde
    Vá ao banheiro
    Escove os dentes
    Tome banho
    Vá ao quarto
    Vista o uniforme
    Vá até a cozinha
    Tome café
    Vá ao quarto
    Pegue o material escolar
    Vá ao ponto de ônibus
    Peque o ônibus que vai até a escola
Fim

Pseudocódigo 2 - Dia a dia do estudante (Crivo nosso)

Devemos destacar que vários algoritmos podem ser definidos para resolver um mesmo problema. Além disso, é necessário e suficiente que as ações sejam definidas rigorosamente e sem ambigüidade, caso contrário a máquina não responderia devidamente às suas ações.

As repetições são também aplicadas aos pseudocódigos. Neste caso, faz-se necessário o uso de palavras como "repita", ou "enquanto" para designar a repetição dos passos desejados. Se quisermos, por exemplo, descrever os passos para fazer um brigadeiro em casa, teríamos:

Início
    Em uma só panela:
        Coloque 400 ml de leite
        Coloque 100 gramas de chocolate
        Coloque 2 colheres de sopa de açúcar
        Coloque 2 colheres de sopa de manteiga
        Coloque uma lata de leite condensado
    Ligue o fogão
    Coloque a panela no fogo
    Enquanto os ingredientes não desgrudar da panela
        Mexa os ingredientes
    Fim do Enquanto
    Desligue o fogão
    O brigadeiro está pronto
Fim

Pseudocódigo 3 – Fazer um brigadeiro (Crivo nosso)

Os fluxogramas e pseudocódigos indicarão apenas as principais ações a serem executadas por um processador. Eles não contêm explicitamente as instruções do computador onde será processado o programa, dando-nos uma ideia geral do algoritmo.

Os algoritmos apresentados ao longo do nosso trabalho abordarão conceitos e operações da matemática elementar, muitas vezes memorizados por alunos e professores, cuja ênfase é dada apenas ao resultado final. Ao contrário, a nossa metodologia, pautada na construção de algoritmos, dá ênfase ao processo de raciocínio para a obtenção da solução do problema. Estes algoritmos serão representados por fluxogramas e/ou pseudocódigos, os quais, acreditamos, sejam facilmente entendidos por alunos do Ensino Básico.

3.2.3 Alguns Algoritmos da Matemática Básica

Os procedimentos algorítmicos são fundamentais na solução de muitos problemas lógicos, matemáticos, técnicos e científicos, os quais desempenham papel importante no modo de vida atual. Desta forma, o conceito de algoritmo é de grande importância na ciência e, portanto, também o deve ser na educação.

Inúmeras operações aritméticas básicas são efetuadas através de algoritmos. Existe, por exemplo, um procedimento bem definido para a adição e multiplicação de números reais, para a extração de raiz quadrada de um número real, o algoritmo de Euclides para encontrar o máximo divisor comum (mdc), algoritmo para a obtenção do mínimo múltiplo comum (mmc), etc. Nunes (2011, p.1) destaca que "[...] a vantagem de descrever modelos de computação na linguagem matemática está na impossibilidade de leitores interpretarem algoritmos de forma diferente, de forma ambígua, ou ainda, na possibilidade de construir máquinas que executam algoritmos como, por exemplo, os computadores e as calculadoras".

3.2.3.1 Algoritmo da Multiplicação

Multiplicar dois números equivale a somar sucessivamente um deles, o mesmo número de vezes do outro, ou seja: $A \times B = A + A + A + \cdots + A$ (B vezes).

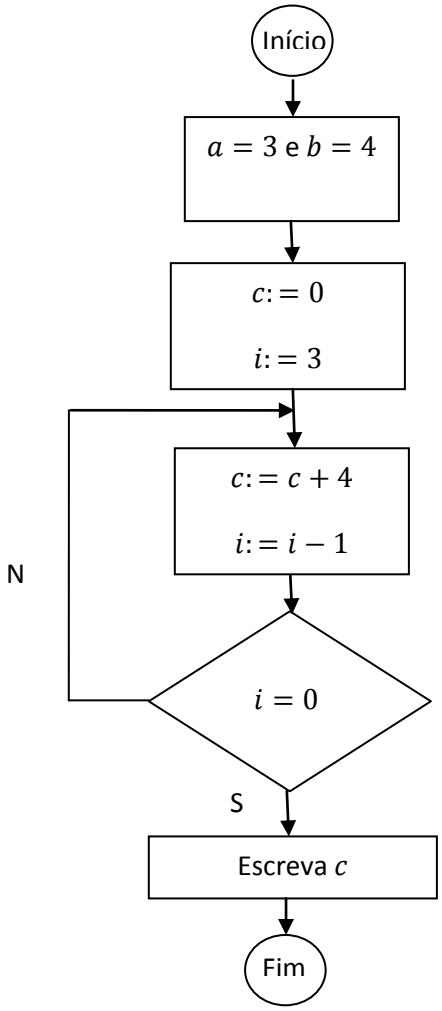
Suponha que estamos interessados em calcular o produto entre dois números naturais a e b, e vamos denotar por c, o produto entre esses dois números. Ou seja, $a \times b = c$.

Considere que o processador disponível:

(a) Não entende sentenças na linguagem natural.

(b) Não sabe multiplicar, mas, apenas, somar.

Vamos construir um dispositivo (algoritmo), representado no fluxograma 4, que efetua a multiplicação de 3 por 4. A cada processamento (do início $i = 0$) vamos tabelando os valores intermediários de c e i. Então temos:



Fluxograma 4 – Algoritmo da multiplicação

Tabela 3 – Valores intermediários do algoritmo da multiplicação

c	i
0	3
4	2
8	1
12	0

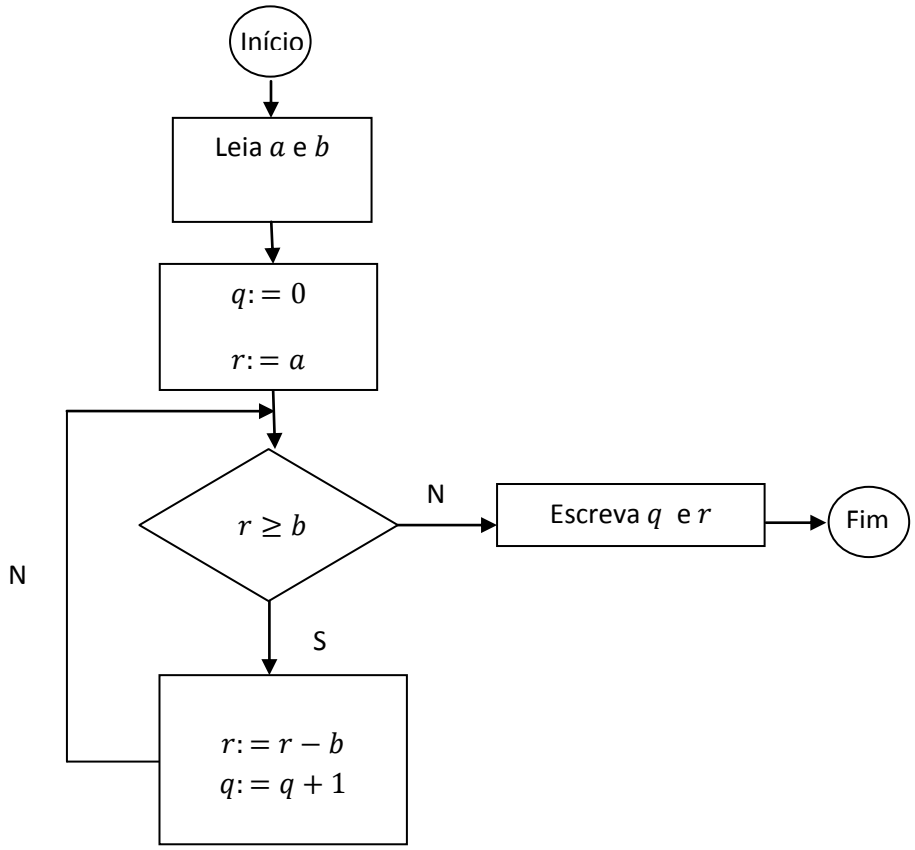
Valor escrito no final do processamento

3.2.3.2 Algoritmo da Divisão de Números Naturais

O quociente da divisão de dois números naturais a e b, com $b < a$ $(a\ /b)$ equivale ao número de vezes que podemos efetuar as subtrações sucessivas: $a - b - b - b - \cdots - b = r$, até que r seja menor do que b $(r < b)$. De fato, pela divisão euclidiana se $b < a$ então, $b = a \cdot q + r$, onde q e r são o quociente e o resto da divisão de a por b, respectivamente.

Por exemplo, podemos afirmar que $15 \div 4 = 3$, pois, $15 - 4 - 4 - 4 = 3$. Note que são feitas 3 subtrações, até que os resultados intermediários sejam maiores que 4. O número 3 equivale ao resto da divisão de 15 por 4.

A divisão de dois números naturais $a\ /b$ pode ser descrita pelo fluxograma 5:



Fluxograma 5- Divisão de números naturais

Início
Leia a e b
$q \coloneqq 0$
$r \coloneqq a$
Repita
$r \coloneqq r - b$
$q \coloneqq q + 1$
Até $r < b$
Escreva q e r
Fim

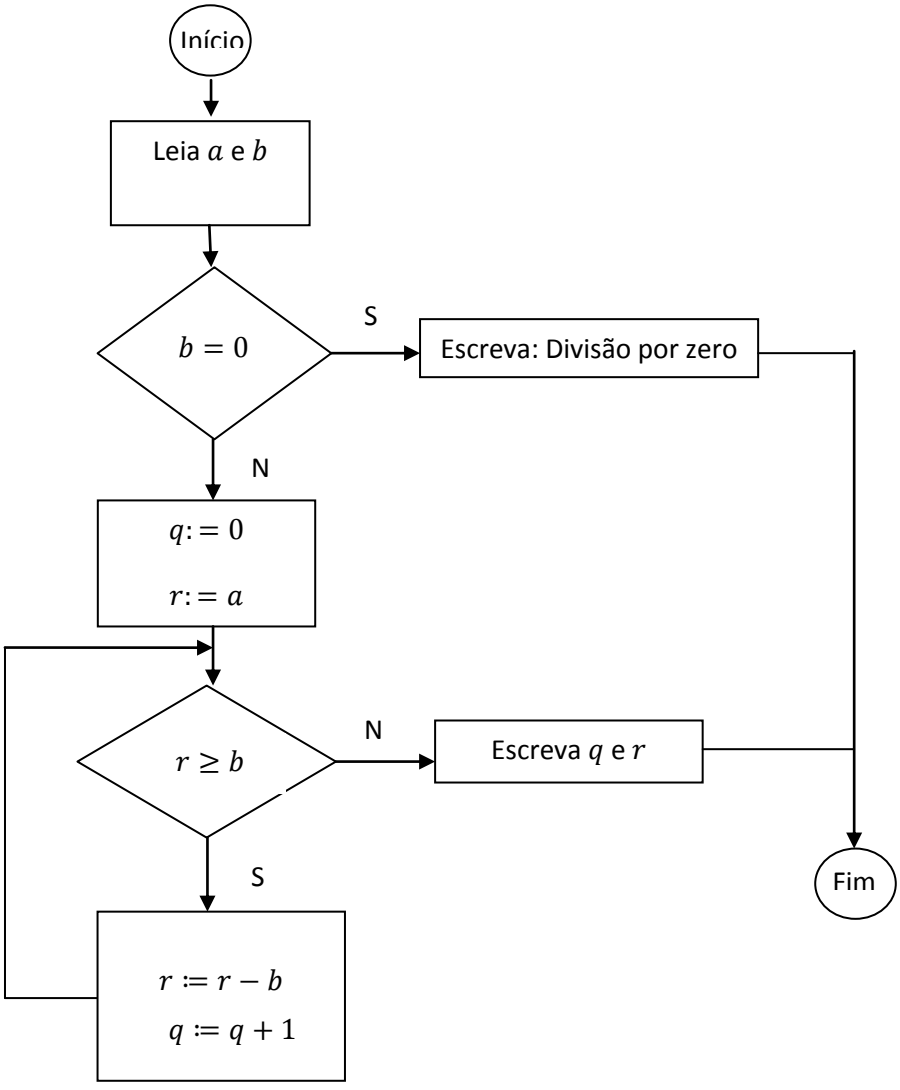
Pseudocódigo 4 – Divisão de números inteiros

No caso de divisões exatas, o resto obtido por subtrações sucessivas é zero. Nada muda no algoritmo descrito no fluxograma 5.

3.2.3.3 Divisão por Zero- Refinamento do Algoritmo da Divisão

No algoritmo da divisão descrito no fluxograma 5, não levou-se em consideração, o caso de $b = 0$. De modo que se formos efetuar a divisão de um número natural a por 0 ($a/0$), utilizando o algoritmo anterior, o processo repetirá infinitamente os comandos internos do bloco de repetição (o algoritmo recairá num *loop* infinito).

Dessa maneira, o processo anterior terá de ser adaptado para esta condição ($b = 0$). Um algoritmo que prevê esta possibilidade é mostrado no fluxograma 6.



Fluxograma 6 – Divisão por zero

Início
    Leia a e b
    Se $b = 0$ *faça*
        Escreva "Divisão por zero"
    Senão
        $q \coloneqq 0$
        $r \coloneqq a$
        Enquanto $r \geq b$ *faça*
            $r \coloneqq r - b$
            $q \coloneqq q + 1$
        Fim do enquanto
        Escreva q e r
    Fim do Senão
Fim

Pseudocódigo 5 – Divisão por zero

3.2.3.4 Máximo Divisor Comum (MDC)

Dados dois números naturais a e b, não simultaneamente nulos, diz-se que o número natural d, diferente de zero, é um divisor comum de a e b se d é divisor simultâneo de a e b, ou seja, se d/a e d/b (dizemos que d divide a e d divide b). Por exemplo, os números 1, 2, 3 e 6 são os divisores comuns de 12 e 18.

O número d é chamado de máximo divisor comum (mdc) de a e b se possuir as seguintes propriedades:

i. d é um divisor comum de a e b, e
ii. d é divisível por todo divisor comum de a e b.

Isto é, o número natural d diferente de zero é o mdc entre dois números inteiros se for o maior elemento da interseção entre os conjuntos dos divisores desses números. Podemos obter este número, aplicando o chamado Algoritmo de Euclides, Hefez (2011, p.56) considera que "[...]este procedimento um primor do ponto de vista computacional, e destaca que pouco conseguiu-se aperfeiçoá-lo em mais de dois milênios". O Algoritmo de Euclides consiste num processo de divisões sucessivas até que se tenha resto zero.

Por exemplo, o mdc entre 12 e 8 é 4 e, seguindo o Algoritmo de Euclides, podemos obtê-lo:

Onde, $q_1, q_2, \dots, q_n$, com $n \in \mathbb{N}^*$, representa o quociente de cada uma das divisões sucessivas, $r_1, r_2, \dots, r_n$, com $n \in \mathbb{N}$, representa o resto dessas respectivas divisões e n equivale ao número de divisões. Note que $\text{mdc}\,(a, b) = r_{n-1}$.

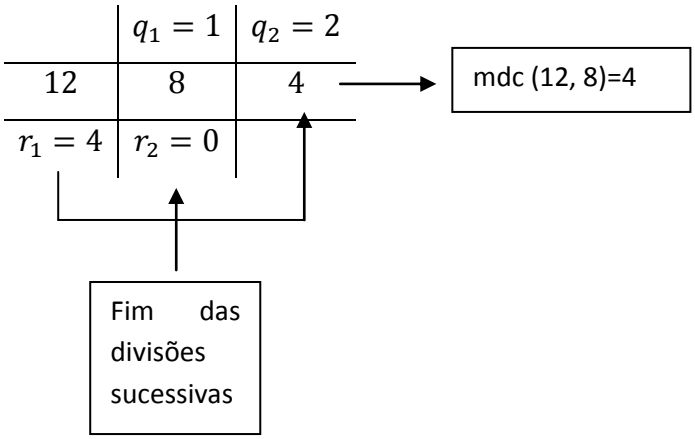
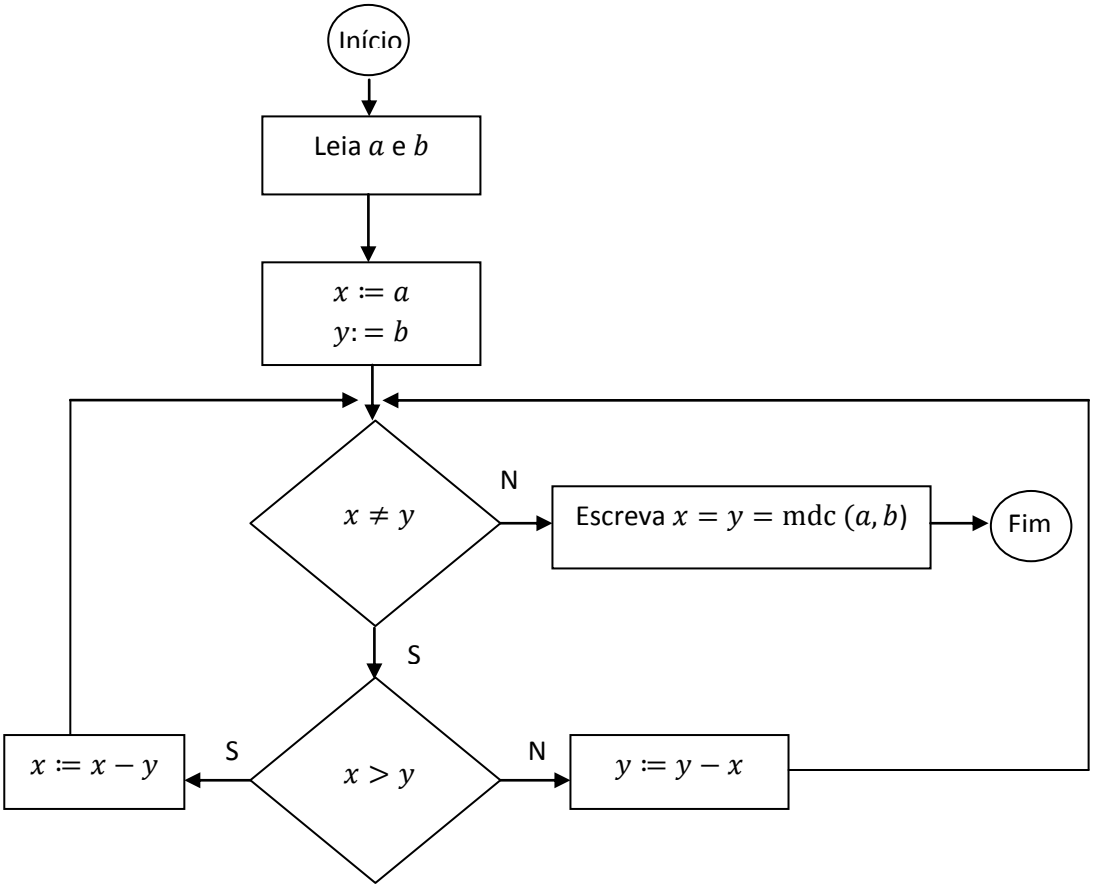


Figura 6 –Algoritmo de Euclides

Como o mdc entre dois números é obtido por divisões sucessivas e a divisão é obtida, como vimos anteriormente, por meio de subtrações sucessivas, podemos determinar o mdc entre dois números naturais a e b através do fluxograma 7 abaixo:



Fluxograma 7 – Máximo Divisor Comum

Início
Leia a e b
$x \coloneqq a$
$y \coloneqq b$
Enquanto $x \neq y$ *faça*
Se $x > y$ *faça*
$x \coloneqq x - y$
Senão
$y \coloneqq y - x$
Fim do senão
Fim do Enquanto
Escreva $x = y = mdc\ (a, b)$

Fim

Pseudocódigo 6 – Máximo Divisor Comum

Para verificar a validade do nosso algoritmo, vamos efetuar o teste na tabela 4, para determinar o mdc (12, 8).

Tabela 4 – Validando algoritmo do MDC

$\boldsymbol{Rodada\ (n)}$	$\boldsymbol{x}$	$\boldsymbol{y}$	$\boldsymbol{x \neq y}$?	$\boldsymbol{x > y}$?	$\boldsymbol{x \coloneqq x - y}$	$\boldsymbol{y \coloneqq x - y}$
1	12	8	sim	sim	$12 - 8 = 4$	_____
2	4	8	sim	$não$	_____	$y = 8 - 4$
3	4	4	$não$		mdc $(12,8) = 4$	

3.2.3.5 Sistemas de Numeração

Dado um conjunto qualquer, podemos contar o número de elementos desse conjunto fazendo agrupamentos. O agrupamento destes elementos pode ocorrer, 2 a 2, neste caso, dizemos que a contagem foi feita na base dois; 3 a 3, dizemos que a contagem foi feita na base três; 10 a 10, a contagem foi feita na base 10.

No sistema de numeração usual, o sistema decimal, nós usamos os dez dígitos do conjunto $\{0, 1, 2, 3, 4, 5\ 6, 7, 8, 9\}$. Um número maior que 9 (nove) é representado usando uma convenção que atribui um peso a cada algarismo, em função da posição que ele ocupa. Em virtude disso, o sistema é também chamado de sistema posicional.

> O sistema universalmente utilizado pelas pessoas comuns para representar os números naturais é o sistema decimal posicional. Este sistema de numeração, que é uma variante do sistema sexagesimal utilizado pelos babilônios 1700 anos antes de Cristo, foi desenvolvido na China e na Índia. Existem documentos do século VI comprovando a utilização desse sistema. Posteriormente, foi se espalhando pelo Oriente Médio, por meio das caravanas, tendo encontrado grande aceitação entre os povos árabes. A introdução do sistema decimal na Europa foi tardia por causa dos preconceitos da Idade Média. Por exemplo, em um documento de 1299, os banqueiros de Florença condenavam o seu uso. (HEFEZ, 2011, p. 43)

Por exemplo, por causa das posições ocupadas pelos dígitos individuais no número 6903, este numeral tem significado numérico calculado como: $6 \times 10^3 + 9 \times 10^2 + 0 \times 10^1 + 3 \times 10^0$. Note que cada peso equivale a uma potência de base dez, cujo expoente varia da direita para a esquerda, sempre em ordem crescente a partir de zero. No sistema decimal, 10 é a base do sistema. Em um sistema numérico com base n, existem n dígitos e o maior é $n - 1$.

3.2.3.5.1 Sistemas de Binário

O sistema binário utiliza apenas os algarismos do conjunto $\{0,1\}$ para a representação numérica dos demais números. Por exemplo, o número 13 é escrito na base 2 como 1101, pois $(1101)_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = (13)_{10}$, de modo análogo ao sistema decimal.

Computadores eletrônicos ‘digitais’ utilizam representação interna baseada em dígitos binários (bits). Esta representação é imprópria para o uso de seres humanos, devido ao grande número de dígitos normalmente envolvidos, mas é muito adequada para a manipulação de circuitos eletrônicos porque os dois valores 0 e 1 podem ser representados, convenientemente, e de modo confiável, pela presença ou ausência de correntes elétricas, cargas elétricas ou campos magnéticos nos circuitos.

3.2.3.5.2 Conversão entre Números Binários e Decimais

Para converter um número natural n da representação decimal para a binária, devemos efetuar uma sequência de divisões por 2. O número n na forma binária resultará dos restos dessas divisões, reescritos da direita para a esquerda.

Para entendermos melhor, vamos converter o número decimal 19 para o sistema binário, ilustrado na figura 7. Iniciamos efetuando a divisão de 19 por 2 e seguimos dividindo sucessivamente o quociente da divisão anterior por dois, até que este quociente seja igual a zero (0). Em seguida, anotamos os restos das respectivas divisões, da esquerda para a direita.

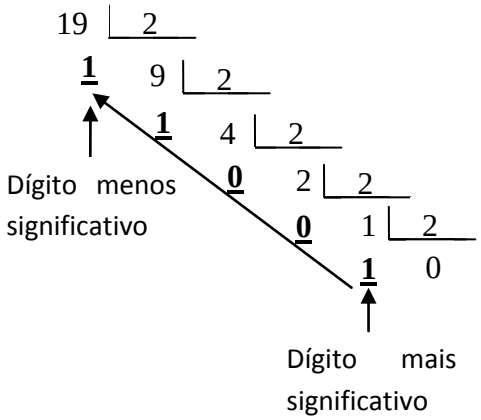


Figura 7 – Conversão para binário

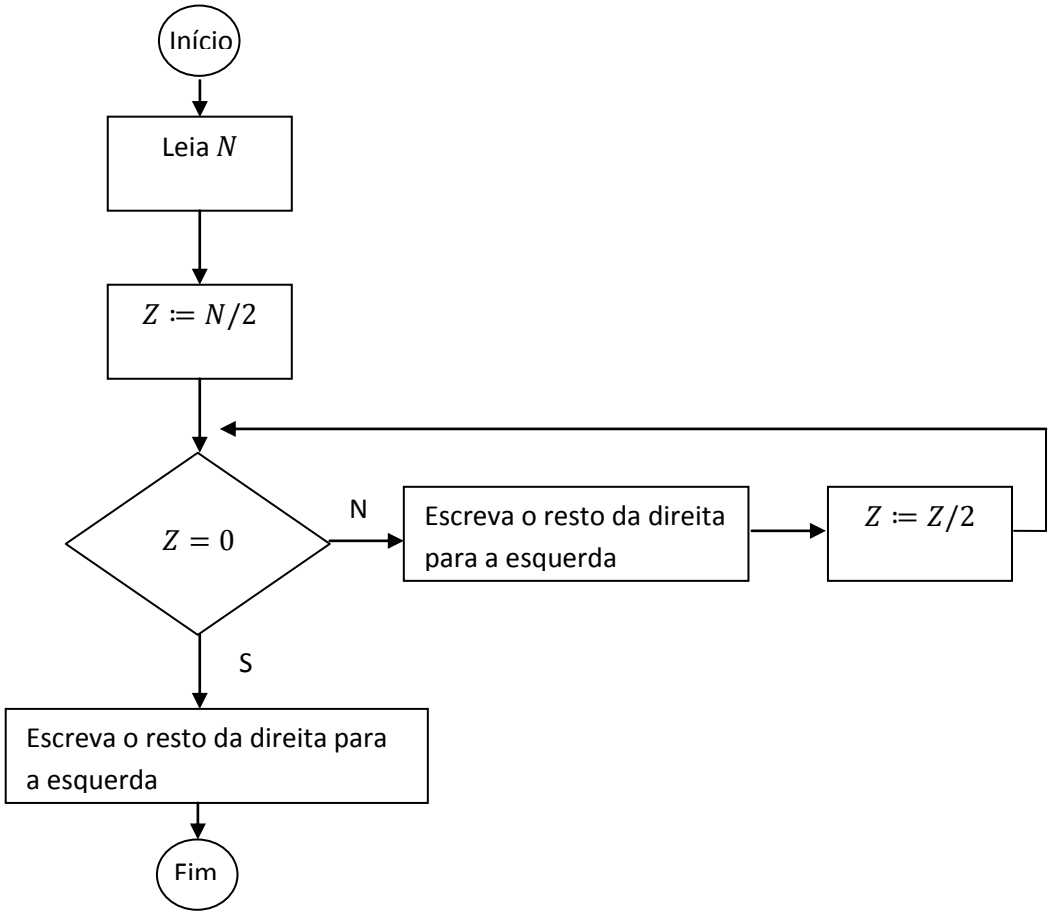
Então, $(19)_{10} = (10011)_2$, ou seja, o número 19 escrito na base decimal equivale ao número 10011 na base binária.

O algoritmo de divisões sucessivas por 2 pode ser ilustrado como no fluxograma 8:

Início
    Leia N
    $Z \coloneqq N/2$
    Enquanto $Z \neq 0$ *faça*
        Escreva o resto da direita para a esquerda
        $Z \coloneqq Z/2$
    Fim do Enquanto
    Escreva o resto da direita para a esquerda

Fim

Pseudocódigo 7 – Conversão de binário para decimal



Fluxograma 8 – Conversão de binário para decimal

Para verificar a validade do nosso algoritmo, vamos efetuar uma simulação, para o caso $N = 13$, na tabela seguinte:

Tabela 5 – Verificando validade do algoritmo de conversão de binário para decimal

Rodada	***Z***	***Z = 0?***	***Resto***	***Z ≔ Z/2***
1	6	*não*	1	3
2	3	*não*	0	1
3	1	*não*	1	0
4	0	*sim*	1	*Fim*

$(13)_{10} = 1\ 1\ 0\ 1$

Números decimais também possuem correspondentes no sistema binário, através de potências negativas de base 2. Vamos aqui considerar o caso em que n é um número decimal entre 0 e 1, portanto a sua representação decimal contém somente 'zero' à esquerda da

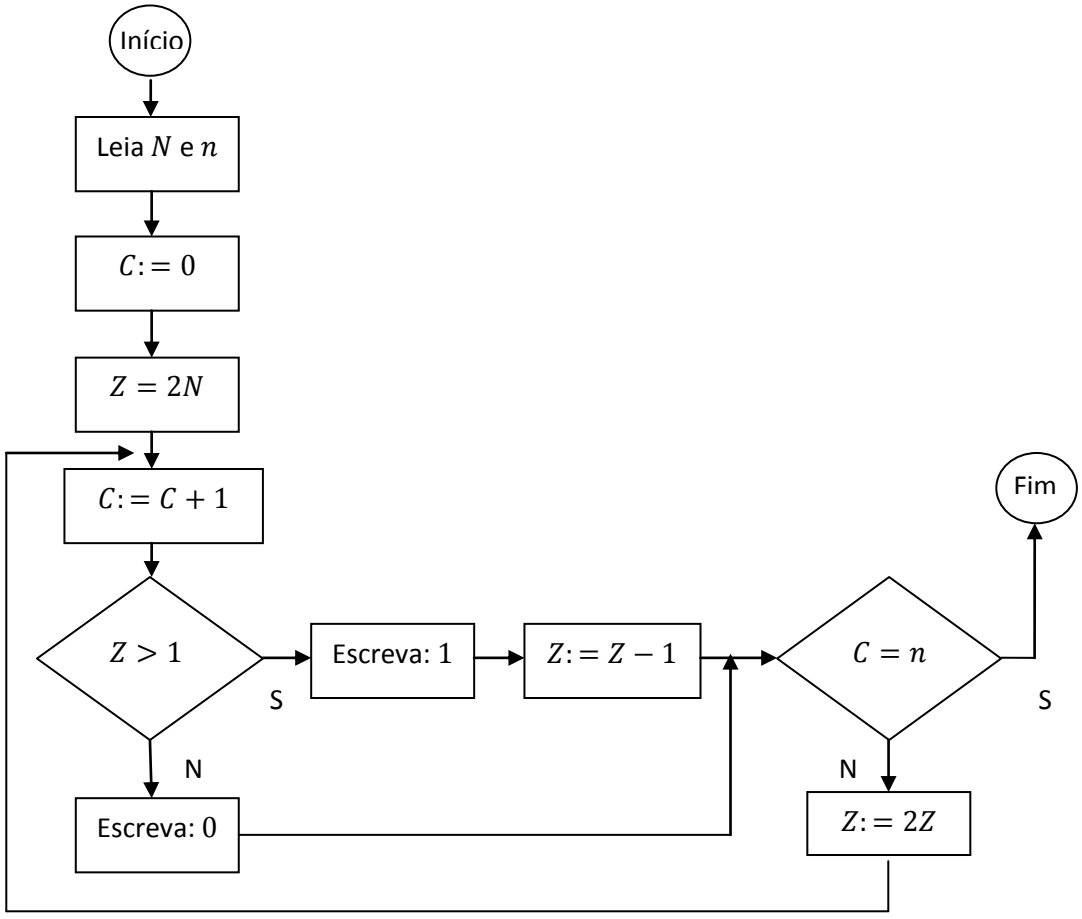
vírgula. A representação binária deste número pode ser encontrada através de sucessivas multiplicações por 2.

A exemplo vamos determinar a representação binária do número 0,69, para até 6 casas decimais.

1. $0{,}69 \times 2 = 1{,}38 \rightarrow 1$: Equivale ao dígito mais significativo
2. $0{,}38 \times 2 = 0{,}76 \rightarrow 0$
3. $0{,}76 \times 2 = 1{,}52 \rightarrow 1$
4. $0{,}52 \times 2 = 1{,}04 \rightarrow 1$
5. $0{,}04 \times 2 = 0{,}08 \rightarrow 0$
6. $0{,}08 \times 2 = 0{,}16 \rightarrow 0$ ...: Equivale ao dígito menos significativo

Portanto, $(0{,}69)_{10} = (0{,}101100\ldots)_2$.

Este algoritmo pode ser descrito pelo fluxograma 9, para n casas decimais.



Fluxograma 9 – Representação binária para *n* casas decimais

Para verificar a validade do nosso algoritmo, vamos efetuar uma simulação, para o caso $N = 0{,}72$, e vamos definir a sua representação binária para 5 casas decimais, depois da vírgula, $n = 5$, como mostra a tabela 6:

Tabela 6 – Validando algoritmo de representação binária para *n* casas decimais

$\boldsymbol{Rodada\ (n)}$	$\boldsymbol{Z}$	$\boldsymbol{C \coloneqq C + 1}$	$\boldsymbol{Z > 1?}$	$\boldsymbol{Nº\ binário}$	$\boldsymbol{Z \coloneqq Z - 1}$	$\boldsymbol{C = n?}$
1	1,44	1	*sim*	1	0,44	*não*
2	0,88	2	*não*	0	____	*não*
3	1,76	3	*sim*	1	0,76	*não*
4	1,52	4	*sim*	1	0,52	*não*
5	1,04	5	*sim*	1	0,04	*sim*

Portanto, $(0{,}72)_{10} = (0{,}101111)_2$ escrito com cinco casas decimais depois da vírgula.

3.2.3.5.3 Os sistemas de Numeração Octal e Hexadecimal

Os sistemas de numeração *octal* e *hexadecimal* são de interesse para nós por sua relação especial com o sistema binário. No sistema octal a base é oito e os dígitos usados pertencem ao conjunto $\{0, 1, 2, 3, 4, 5, 6, 7\}$. No sistema hexadecimal a base é dezesseis, sendo usados os dez dígitos decimais $0, 1, 2, \ldots, 9$ para representar dez dos dígitos necessários, sendo os outros seis normalmente representados pelas letras $\mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D}, \mathrm{E}$ e F.

A relação especial entre os sistemas octal e hexadecimal e os sistemas binário resulta do fato de que três dígitos binários podem representar exatamente oito (2^3) números diferentes e que quatro dígitos binários podem representar dezesseis (2^4) números diferentes. Assim, a conversão de binário para octal ou hexadecimal ou vice-versa é facilmente executada. Para passar de binário para octal, agrupamos os dígitos binários três a três a partir da vírgula binária, em ambas as direções, e substituímos cada grupo por seu equivalente octal. Para passar de binário para o hexadecimal, agrupamos de quatro em quatro.

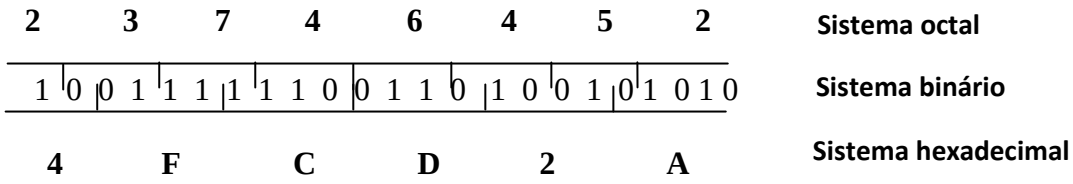


Figura 8 – Sistemas de numeração

3.2.3.5.4 Operações de adição

O algoritmo utilizado para somar dois números binários é semelhante ao utilizado na base decimal. Ou seja, é baseado no fato de transferirmos unidades para o agrupamento de ordem imediatamente superior. Por exemplo, no sistema decimal, $9 + 3 = 12$ – significa 1 unidade de agrupamentos de 10 somado com 2 unidades de agrupamentos de 1 – no sistema binário, $1 + 1 = 10$ – significa que 1 unidade foi transferida à ordem seguinte, o que equivale dizer que que $1 + 1 = 0$ (vai um), pois não temos neste caso outro algarismo para representar a quantidade 'dois'. Dessa maneira, se o número é maior que o maior algarismo da base, somamos unidades de agrupamentos à posição seguinte (ordem superior).

$1 + 1 = 0$ (com a soma de 1 na posição seguinte)
$1 + 0 = 1$
$0 + 0 = 0$

Exemplo:

```
 1 1 1 1  ⟶  Transferidos para
   1 0 1 1     a posição seguinte
+  1 1 1 1     (vai um)
 1 1 0 1 0
```

3.2.3.5.5 Operação de Subtração

De acordo com Lima (2010, p.204), para efetuar a subtração no sistema binário, devemos "[...] levar em conta a ordem das parcelas, por não ser uma operação comutativa[...]". Analogamente ao sistema decimal, tem-se *minuendo x, subtraendo y, diferença D* e o *empréstimo E*.

Primeiramente, vamos recordar que: (a) toda vez que um dígito minuendo for menor que o correspondente dígito subtraendo, toma-se 1 unidade emprestada ao dígito posicionado imediatamente à esquerda; (b) os números 1 e 10, no sistema binário equivalem a 1 e 2 no sistema decimal, respectivamente.

Exemplo:

0 0 1 1 ⟶ Empréstimo *E*
1 1 1 0 ⟶ Minuendo *x*
-1 0 1 1 ⟶ Subtraendo *y*
0 0 1 1 ⟶ Diferença *D*

3.3 Sequências, Ordenações e Buscas

Um computador eletrônico possui um sistema de memórias para guardar e manipular dados. A memória principal do computador (memória RAM) é um tipo de memória que manipula os dados e armazena as informações do programa corrente. Esta memória não é permanente, então, se desligarmos o computador perderemos os dados gravados. A memória secundária (disco rígido) é comparável a um 'arquivo', e grande quantidade de dados podem ser guardados permanentemente nela.

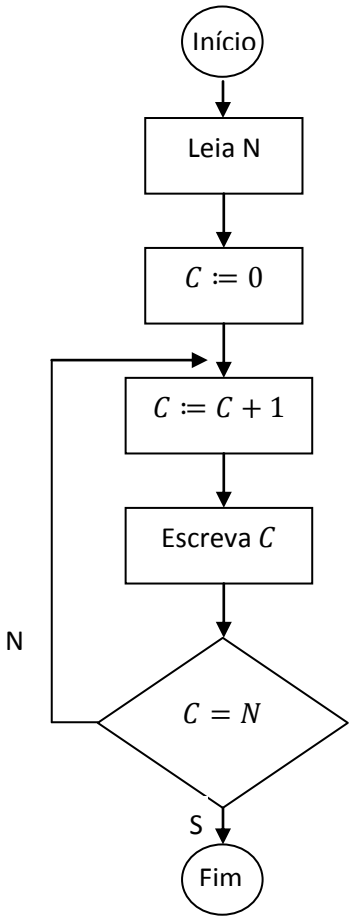
Para gravarmos dados na memória principal podemos utilizar estruturas chamadas vetores, que são variáveis indexadas, como por exemplo $A[I] = A[1], A[2], \ldots$ que podem ser recuperados aleatoriamente, ou seja, podemos recuperar qualquer sequência $A[I], 1 < I < k$, sem percorrê-la, elemento por elemento. Os dados gravados em memória secundária só poderão ser recuperados sequencialmente.

3.3.1 Sequências

Sequências são conjuntos de caracteres (números, letras, etc.) que podem ser gravados em memória de computador. Algumas sequências numéricas podem ser obtidas e gravadas através de algoritmos. Vejamos alguns exemplos:

3.3.1.1 Sequência de Números Naturais de 1 a N

Esta sequência pode ser obtida e escrita através do algoritmo apresentado no fluxograma 10:



Fluxograma 10 – Sequência de números naturais

Início
    Leia N
    $C \coloneqq 0$
    Enquanto $C \neq N$ *faça*
        $C \coloneqq C + 1$
        Escreva C
    Fim do Enquanto
Fim

Pseudocódigo 8 – Sequencia de números naturais

Para verificar a validade do nosso algoritmo, vamos efetuar uma simulação, para o caso $N = 5$, como mostra a tabela 7:

Tabela 7 – Verificar validade do algoritmo de sequência dos números naturais

$\boldsymbol{Rodada\ (n)}$	$\boldsymbol{C \coloneqq C + 1}$	$\boldsymbol{C\ gravado}$	$\boldsymbol{C = N?}$
1	1	1	$não$
2	2	2	$não$
3	3	3	$não$
4	4	4	$não$
5	5	5	sim

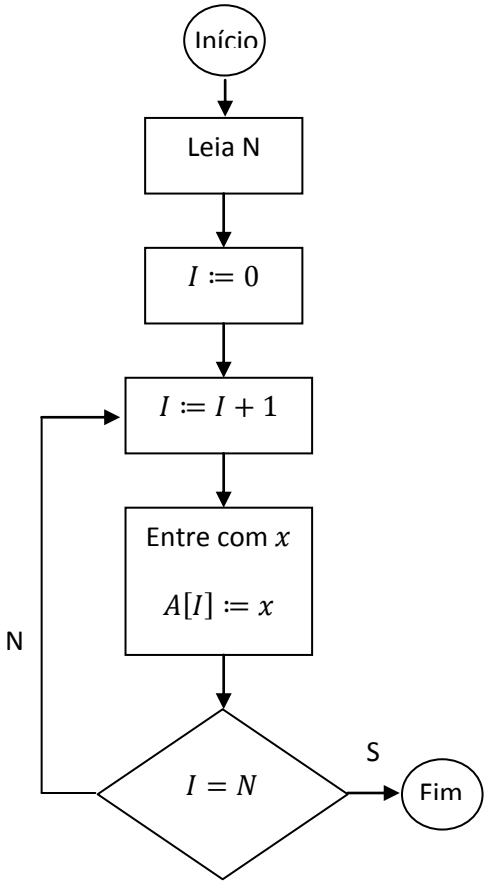
3.3.1.2 Sequência I de 1 a N

Uma sequência I de 1 a N pode servir de índice para uma variável indexada $A[I] = A[1], A[2]$,..., $A[N]$ chamada vetor.

Podemos gravar na memória de um computador N valores $A[I]$ com $1 \leq I \leq N$. O fluxograma 11 descreve um algoritmo que armazena uma sequência de N números naturais:

Início
    Leia N
    $I \coloneqq 0$
    Enquanto $I \neq N$ *faça*
        $I \coloneqq I + 1$
        Escreva X
        $A[I] \coloneqq X$
    Fim do Enquanto
Fim

Pseudocódigo 9 – Sequência I de 1 a *n*



Fluxograma 11 – Sequência I de 1 a *n*

Vamos supor que estamos interessados em armazenar os 5 algarismos e efetuar a simulação abaixo, na tabela 8:

Tabela 8 – Verificação do algoritmo da sequência I de 1 a *n*

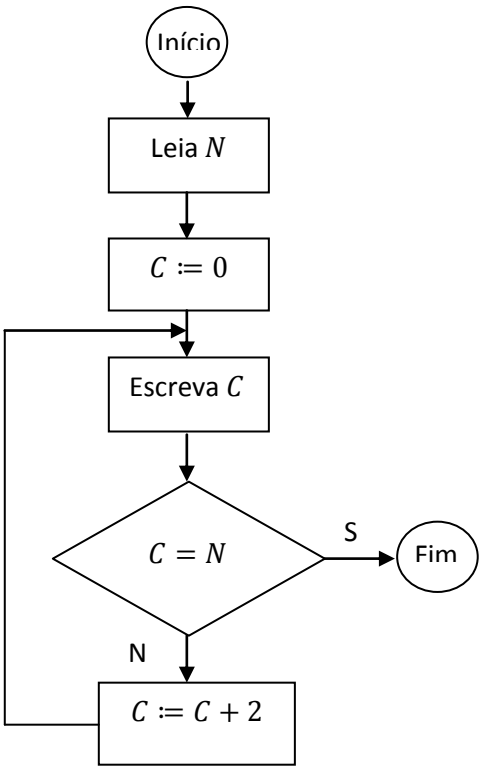
$\boldsymbol{Rodada}$	$\boldsymbol{I \coloneqq I + 1}$	$\boldsymbol{x}$	$\boldsymbol{A[I]}$	$\boldsymbol{I = N?}$
1	1	7	7	$não$
2	2	4	4	$não$
3	3	5	5	$não$
4	4	12	12	$não$
5	5	17	17	sim

Pelo algoritmo acima, fica gravado na memória: $A[1] = 7$; $A[2] = 4$; $A[3] = 5$; $A[4] = 12$; $A[5] = 17$.

Obs.: Os valores de X podem ser digitados pelo teclado do computador.

3.3.1.3 Sequência de Números Pares de 0 a N

Sequências de números pares de 0 a N podem ser geradas e escritas pelo algoritmo do fluxograma 12:



Fluxograma 12 – Sequência de números pares

Início
    Leia N
    $C \coloneqq 0$
    Enquanto $C \neq N$ *faça*
        Escreva C
        $C \coloneqq C + 2$
    Fim do Enquanto
Fim

Pseudocódigo 10 – Sequência de números pares

Para gravarmos a sequência de pares de 0 a 10, temos que fazer $N = 10$ e seguir o fluxograma, anotando os respectivos valores de C gravados em cada rodada.

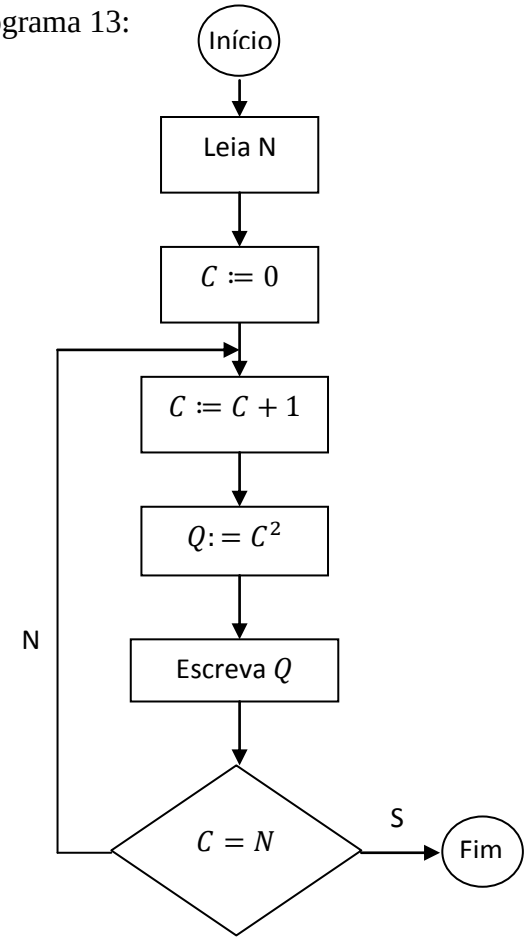
Tabela 9 – Verificando algoritmo da sequência de números pares

$\boldsymbol{Rodada}$	$\boldsymbol{C\ gravado}$	$\boldsymbol{C = N?}$	$\boldsymbol{C \coloneqq C + 2}$
1	0	$não$	2
2	2	$não$	4
3	4	$não$	6
4	6	$não$	8
5	8	$não$	10
6	10	sim	Fim

Sequência: $0, 2, 4, 6, 8, 10$.

3.3.1.4 Sequência de Quadrados Perfeitos

A sequência de quadrados perfeitos: $1, 4, 9, 16, 25, \ldots, n^2$, pode ser obtida pelo seguinte algoritmo apresentado no fluxograma 13:



Fluxograma 13 – Sequência de quadrados perfeitos

Início
    Leia N
    $C \coloneqq 0$
    Enquanto $C \neq N$ *faça*
        $C \coloneqq C + 1$
        $Q \coloneqq C^2$
        Escreva Q
    Fim do Enquanto
Fim

Pseudocódigo 11 – Sequência de quadrados perfeitos

Vamos testar a rotina do fluxograma 13, gravando os 5 primeiros quadrados perfeitos, para isso, vamos considerar $N = 5$ e anotar os valores de Q gravados, como mostra a tabela 10:

Tabela 10 – Verificando validade do algoritmo da sequência de quadrados perfeitos

$\boldsymbol{Rodada}$	$\boldsymbol{C \coloneqq C + 1}$	$\boldsymbol{Q = C^2}$	$\boldsymbol{C = N?}$
1	1	1	$não$
2	2	4	$não$
3	3	9	$não$
4	4	16	$não$
5	5	25	sim

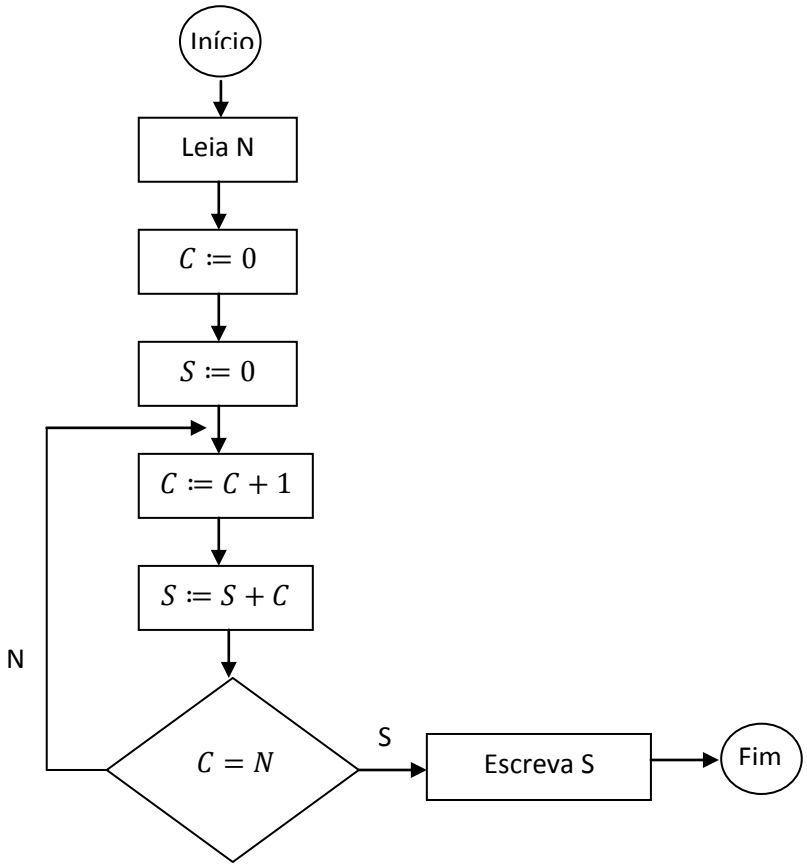
3.3.1.5 Operações com Termos de uma Sequência

Podemos construir algoritmos que nos possibilita operar com os termos de uma sequência e escrever o resultado, assim como ler números gravados em memória e operá-los.

A rotina do fluxograma 14, estabelece a soma dos termos de uma sequência de números naturais, descritos de 1 a N:

Início
    Leia N
    $C \coloneqq 0$
    $S \coloneqq 0$
    Enquanto $C \neq N$ *faça*
        $C \coloneqq C + 1$
        $S \coloneqq S + C$
    Fim do Enquanto
    Escreva S
Fim

Pseudocódigo 12 – Soma dos termos de uma sequência



Fluxograma 14 – Soma dos termos de uma sequência

Vamos somar os 5 primeiros números naturais utilizando o algoritmo anterior, para isso, consideremos $N = 5$. Veja a tabela 11:

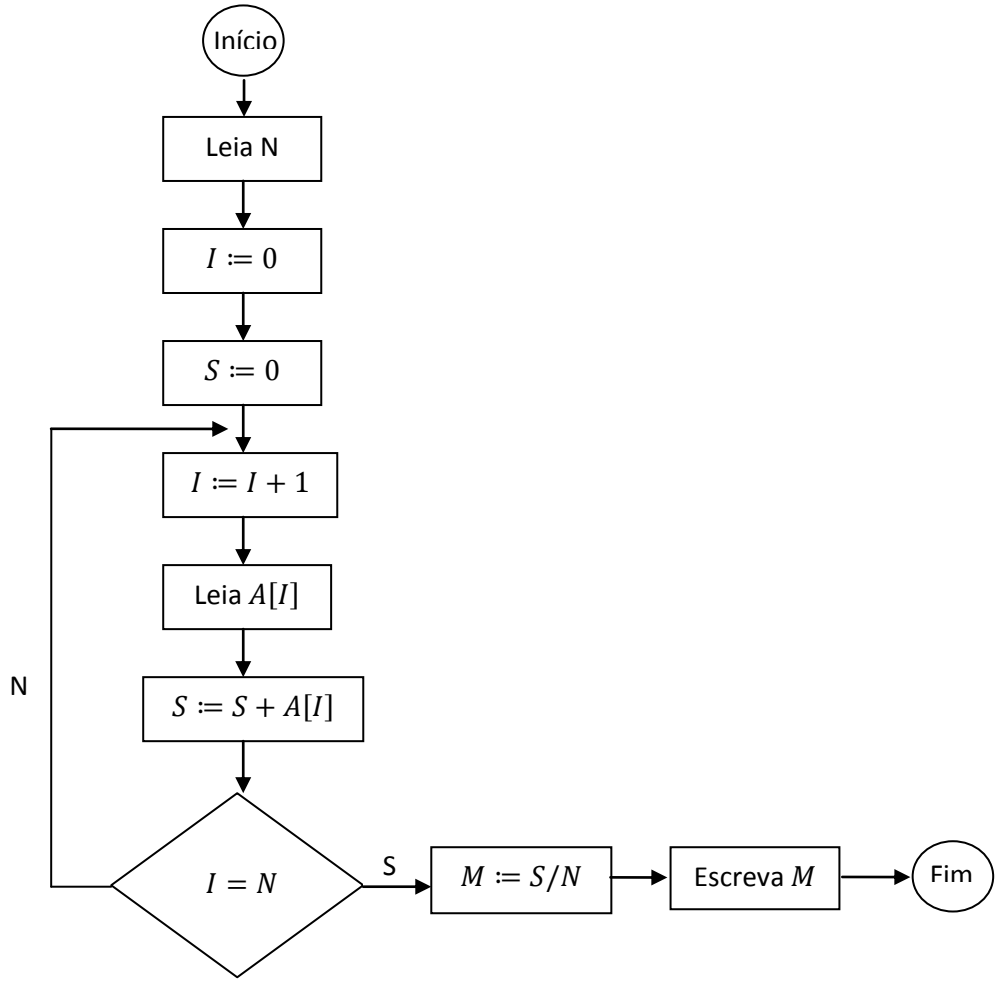
Tabela 11 – Verificando validade do algoritmo da soma dos termos de uma sequência

$\boldsymbol{Rodada}$	$\boldsymbol{C \coloneqq C + 1}$	$\boldsymbol{S \coloneqq S + C}$	$\boldsymbol{C = N?}$
1	1	1	$não$
2	2	3	$não$

3	3	6	$não$
4	4	10	$não$
5	5	15	sim

Para recuperar uma sequência de números gravados em memória, operando com seus termos, fazemos a cada rodada a leitura de um termo da sequência e operamos com este termo usando um acumulador.

Assim, se temos N números gravados numa variável $A[I]$ com $1 \leq I \leq N$, podemos, por exemplo, recuperar esta sequência e calcular a média aritmética (M) dos seus termos, como mostra o fluxograma 15:



Fluxograma 15 – Calculando a média aritmética de uma sequência

Início
    Leia N
    $I \coloneqq 0$
    $S \coloneqq 0$
    Enquanto $I \neq N$ *faça*
        $I \coloneqq I + 1$
        Leia $A[I]$
        $S \coloneqq S + A[I]$
    Fim do Enquanto
    $M \coloneqq S/N$
    Escreva M
Fim

Pseudocódigo 13 – Média aritmética de uma sequência

Supondo $A[1] = 10, A[2] = 5, A[3] = 3$, vamos recuperar estes números e calcular a respectiva média aritmética. Veja a tabela 12:

Tabela 12 – Verificando a validade do algoritmo da média aritmética de uma sequência

$\boldsymbol{Rodada}$	$\boldsymbol{I \coloneqq I + 1}$	$\boldsymbol{A[I]}$	$\boldsymbol{S \coloneqq S + A[I]}$	$\boldsymbol{I = N}$?	$\boldsymbol{M \coloneqq S/N}$
1	1	10	10	$não$	_____
2	2	5	15	$não$	_____
3	3	3	18	sim	6

3.3.2 Ordenações

Dados podem ser lidos da memória e escritos numa certa ordem que nos interessa. Para isso temos de aplicar algoritmos que possam fazer com que estes dados possam ser-nos apresentado na ordem desejada.

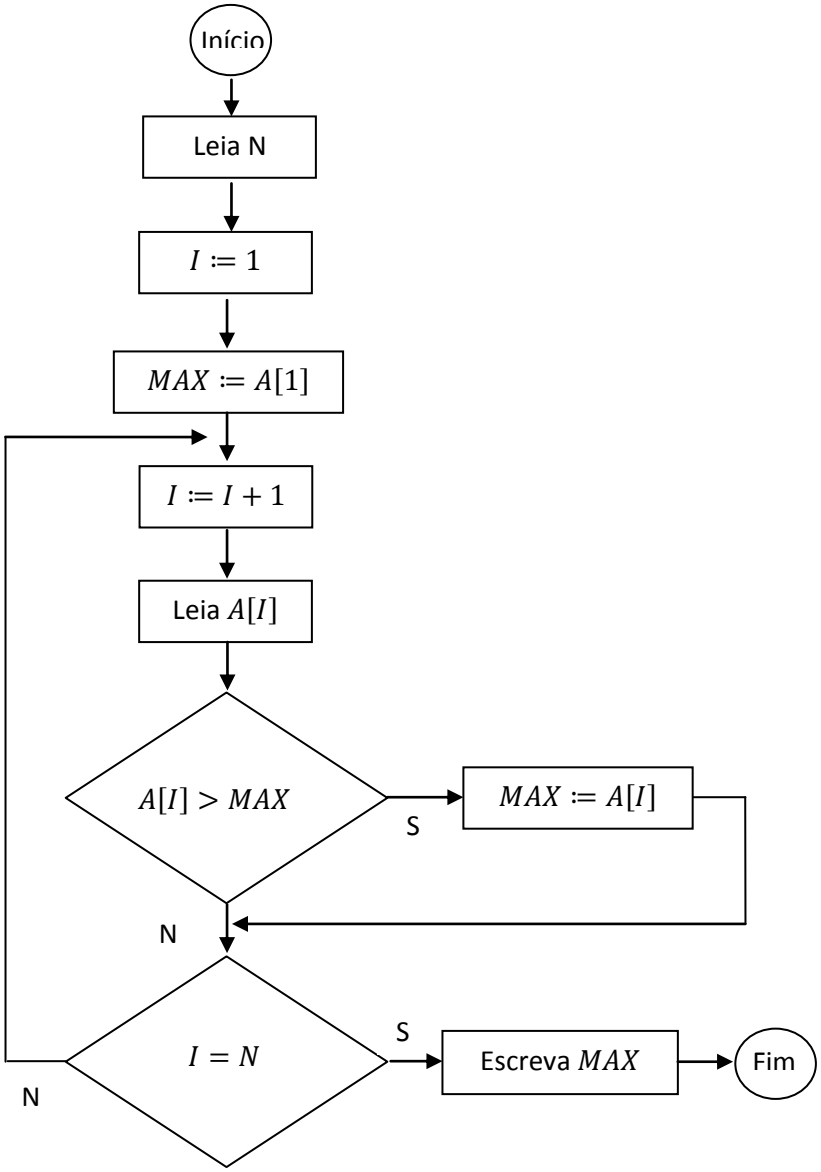
Antes, porém, teremos de estudar algoritmos básicos para a ordenação, como por exemplo, algoritmos que possam escolher o **maior** e o **menor** termo de uma sequência numérica.

3.3.2.1 Termo Máximo e Mínimo

Para selecionarmos o termo máximo ou maior termo de uma sequência com N termos, com $N \geq 2$, podemos utilizar a rotina representada no fluxograma 16:

Início
    Leia N
    $I \coloneqq 1$
    $MAX \coloneqq A[1]$
    Enquanto $I \neq N$ *faça*
        $I \coloneqq I + 1$
        Leia $A[I]$
        Se $A[I] > MAX$ *faça*
            $MAX \coloneqq A[I]$
        Fim do Se
    Fim do Enquanto
    Escreva MAX
Fim

Pseudocódigo 14 – Termo máximo



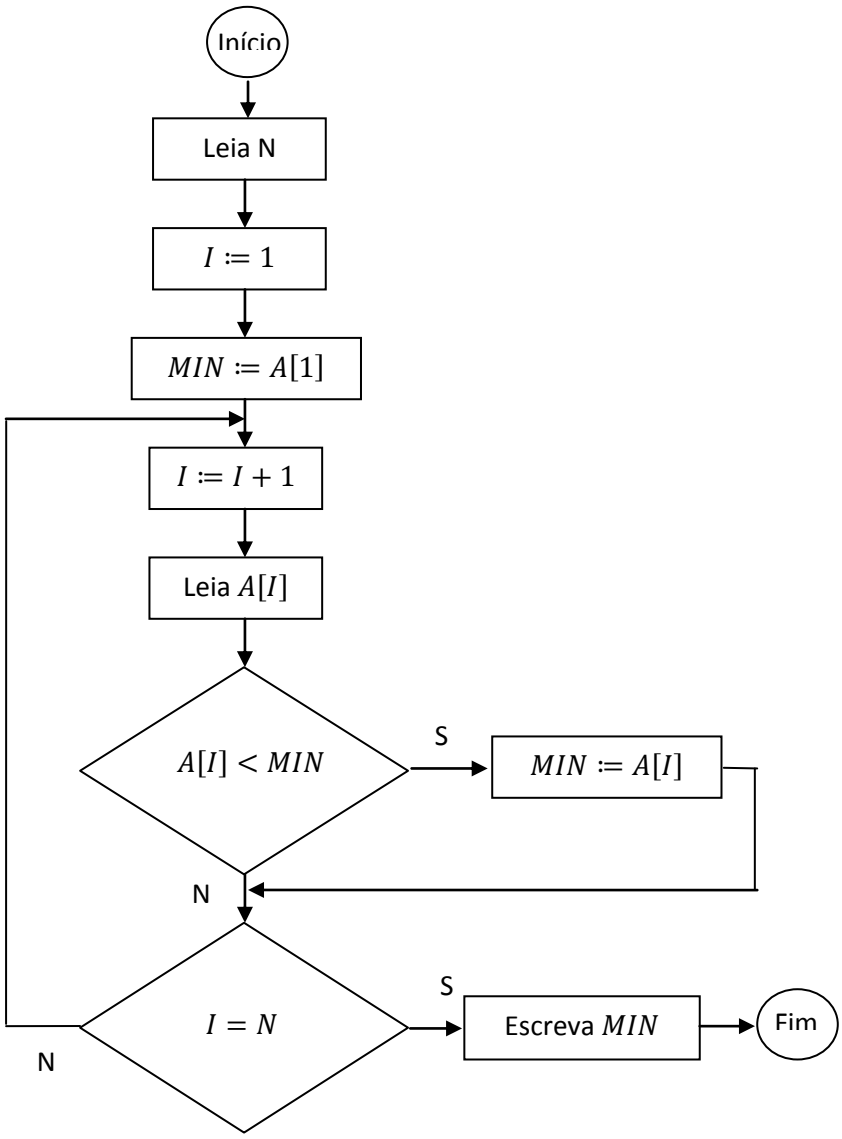
Fluxograma 16 – Termo máximo

Vamos supor que estamos interessados em determinar o maior termo da sequência: $A[1] = 10, A[2] = 7, A[3] = 15, A[4] = 20$. Então, para $N = 4$, temos:

Tabela 13 – Verificando validade do algoritmo do termo máximo

$Rodada$	$MAX \coloneqq A[I]$	$I \coloneqq I + 1$	$A[I]$	$A[I] > MAX?$	$MAX: = A[I]$	$I = N?$
1	10	2	7	$não$	_____	$não$
2	_____	3	15	sim	15	$não$
3	_____	4	20	sim	20	sim

Para selecionarmos o termo mínimo de uma sequência de N termos, com $N \geq 2$ termos, podemos utilizar um algoritmo análogo ao anterior. Observe o fluxograma 17:



Fluxograma 17 – Termo mínimo

Início
    Leia N
    $I \coloneqq 1$
    $MIN \coloneqq A[1]$
    Enquanto $I \neq N$ *faça*
        $I \coloneqq I + 1$
        Leia $A[I]$
        Se $A[I] < MIN$ *faça*
            $MIN \coloneqq A[I]$
        Fim do Se
    Fim do Enquanto
    Escreva MIN
Fim

Pseudocódigo 15 – Termo mínimo

Vamos supor que estamos interessados em determinar o menor termo da sequência: $A[1] = 10, A[2] = 7, A[3] = 15, A[4] = 20$. Então, para $N = 4$, temos:

Tabela 14 – Verificando validade do algoritmo do termo mínimo

$\boldsymbol{Rodada}$	$\boldsymbol{MIN \coloneqq A[I]}$	$\boldsymbol{I \coloneqq I + 1}$	$\boldsymbol{A[I]}$	$\boldsymbol{A[I] < MIN?}$	$\boldsymbol{MIN := A[I]}$	$\boldsymbol{I = N?}$
1	10	2	7	sim	7	$não$
2		3	15	$não$	_____	$não$
3		4	20	$não$	_____	sim

3.3.2.2 Ordenação por Seleção Direta

O método de ordenação por seleção é baseado no seguinte princípio:

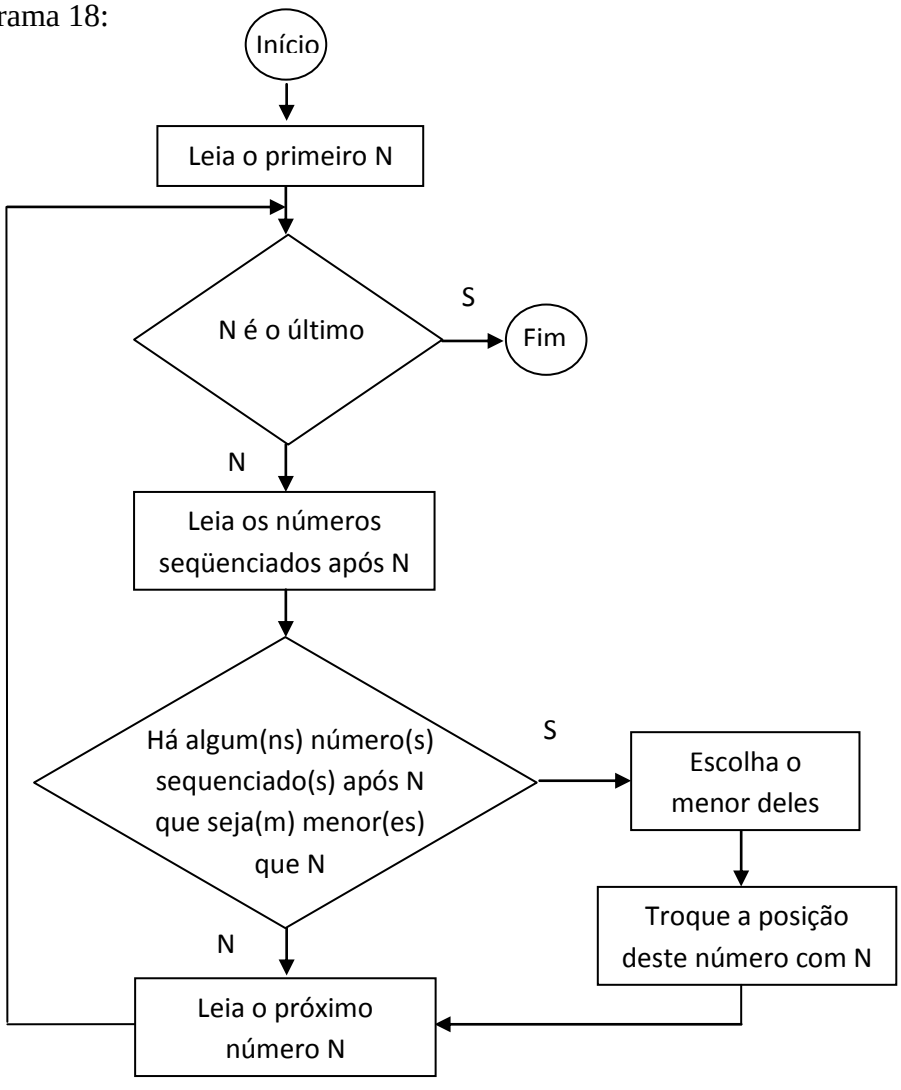
1. Seleciona-se o termo de menor valor (mínimo);
2. Troca-o com o primeiro elemento da sequência $A[I]$;
3. Repete-se as operações anteriores, envolvendo apenas os $N - 1$ termos restantes, depois envolvendo os $N - 2$ restantes e assim sucessivamente, até restar apenas um dos elementos, o maior deles.

Vamos aplicar o método anterior para ordenar os números: $44, 55, 12, 42, 94, 18, 06, 67$. Como mostra a figura 9:

44	55	12	42	94	18	06	67	→ Sequência inicial
06	55	12	42	94	18	**44**	67	→ 1ª Rodada
06	**12**	**55**	42	94	18	44	67	→ 2ª Rodada
06	12	**18**	42	94	**55**	44	67	→ 3ª Rodada
06	12	18	**42**	94	55	44	67	→ 4ª Rodada
06	12	18	42	**44**	55	**94**	67	→ 5ª Rodada
06	12	18	42	44	**55**	94	67	→ 6ª Rodada
06	12	18	42	44	55	**67**	**94**	→ 7ª Rodada

Figura 9 – Ordenando número por seleção direta

Um algoritmo para ordenar os números N de uma sequência de números dados pode ser descrito pelo fluxograma 18:



Fluxograma 18 – Ordenação por seleção direta

3.3.3 Buscas

As buscas são tarefas frequentemente encontradas em programação de computadores. Há uma grande variedade de algoritmos desenvolvidos com esta finalidade. Em geral, a busca consiste em encontrar numa sequência de dados $A[I]$, um dado que nos interessa, como encontrar uma ficha num arquivo ou uma palavra no dicionário.

3.3.3.1 Busca Linear

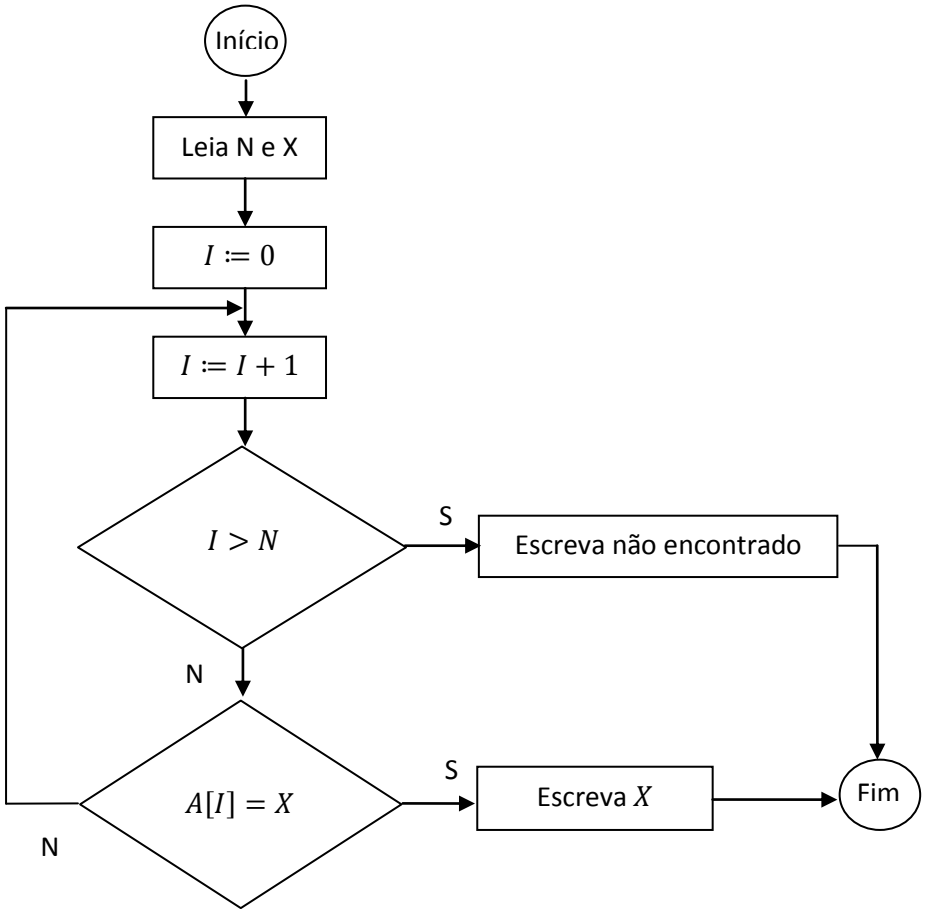
Nos casos em que não se dispõe de informações adicionais sobre os dados a serem buscados, usamos a busca seqüencial por toda a sequência (vetor).

A busca termina quando:

(a) O elemento é encontrado $(A[I] = X)$;

(b) Toda a sequência (vetor) foi analisada, mas, o elemento não foi encontrado.

Em uma sequência de N elementos podemos usar o algoritmo apresentado no fluxograma 19, para encontrar o elemento X:



Fluxograma 19 – Busca linear

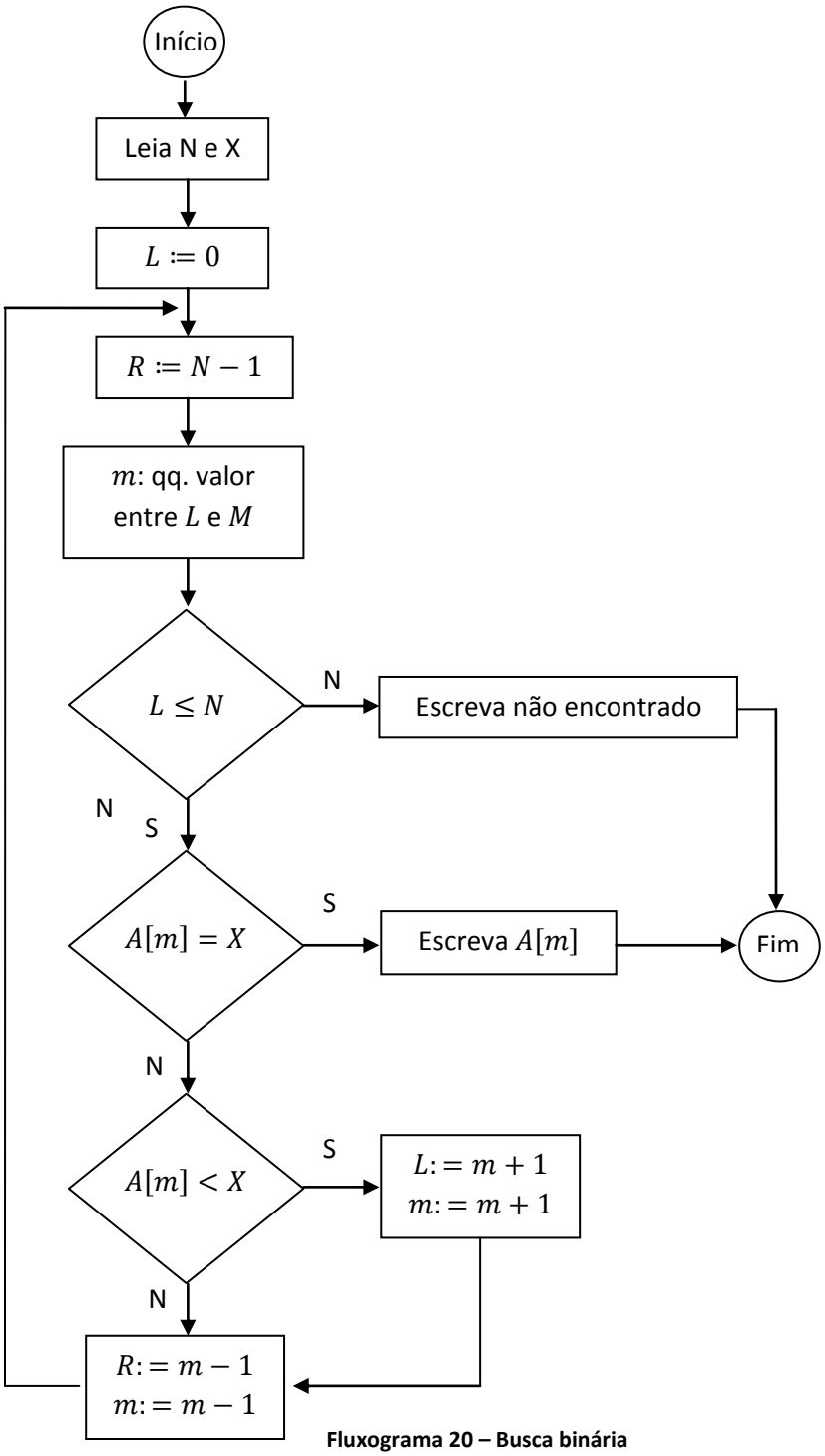
Supondo a sequência em que $A[1] = 2, A[2] = 4, A[3] = 7, A[4] = 5, A[5] = 50$, vamos localizar o elemento $X = 5$, como mostra a tabela 15:

Tabela 15 – Verificando validade do algoritmo da busca linear

$\boldsymbol{Rodada}$	$\boldsymbol{I \coloneqq I + 1}$	$\boldsymbol{I > N?}$	$\boldsymbol{A[I] = X?}$
1	1	$não$	$não$
2	2	$não$	$não$
3	3	$não$	$não$
4	4	$não$	sim

3.3.3.2 Busca Binária

Não existem meios de acelerar uma busca sempre que se tem maiores informações sobre o elemento a ser localizado. Uma busca pode ser mais eficaz se os dados estiverem ordenados. A seguir apresentaremos um algoritmo com base em dados ordenados, ou seja, cujos dados são elementos de uma sequência $A[I]: 1 \leq I \leq N, A[I-1] \leq A[I]$.



Fluxograma 20 – Busca binária

A ideia principal é a de testar um elemento sorteado aleatoriamente, um $A[m]$qualquer, com m pertencente ao intervalo $[1, N]$ e compará-lo com o argumento de busca X. Se tal elemento for igual a X a busca termina, se for menor que X, conclui-se que todo elemento com índice menor ou igual a m pode ser eliminado, se for maior que X, todos aqueles elementos com índice maior ou igual a m pode também ser eliminado. Isto resulta no algoritmo chamado de busca binária: este algoritmo utiliza duas variáveis índices L e R, que marcam os limites esquerdo e direito da região de $A[I]$ em que o elemento pode ser encontrado.

Este algoritmo é análogo à procura de uma palavra no dicionário. Primeiro abrimos o dicionário aleatoriamente e verificamos se a palavra encontrada está à esquerda ou à direita. Depois poderemos eliminar uma das partes e procuramos página por página.

O fluxograma 20 trata de um algoritmo que efetua a busca binária, o qual deseja-se encontrar um elemento X na sequência $A[I]$, com I pertencente ao intervalo $[1, N]$. Para verificar a sua validade, vamos supor que na sequência ordenada $A[I]: 1, 3, 5, 7, 9, 11, 13, 15$, estamos interessados em determinar o elemento $X = 7$. Então temos para $N = 8$:

Tabela 16 – Verificando validade do algoritmo de busca binário

$\boldsymbol{Rodada}$	$\boldsymbol{L}$	$\boldsymbol{R}$	$\boldsymbol{m}$	$\boldsymbol{A[m]}$	$\boldsymbol{L \leq N}$	$\boldsymbol{A[m] = X?}$	$\boldsymbol{A[m] < X}$
1	0	7	7	13	*sim*	*não*	*não*
2	0	6	6	11	*sim*	*não*	*não*
3	0	5	5	9	*sim*	*não*	*não*
4	0	4	4	7	*sim*	*sim*	*Fim*

4. SUGESTÕES DE ATIVIDADES

Os algoritmos são trabalhados com alunos no Ensino Básico, desde o momento em que eles iniciam o seu primeiro contanto com a Matemática. Neste capítulo, sugerimos quatro atividades que podem ser aplicadas em sala de aula, como estímulo ao raciocínio lógico e ao pensamento algorítmico e computacional. A natureza e objetivos das atividades é que

determinará a partir de qual nível escolar os estudantes poderão ser estimulados a desenvolvê-las.

Podemos elaborar atividades de execução, análise e/ou construção de algoritmos que podem ser aplicadas a alunos do 2º ano do ensino fundamental ao 3º ano do ensino médio e requerem o desenvolvimento de competências e habilidades como autonomia, capacidade de análise, de trabalhar em grupo, criatividade, auto de grau de compreensão sobre os conceitos básicos de linguagem de programação e de matemática, entre outros. São atividades simples, com o objetivo de revisar conhecimentos prévios, operações básicas e de introduzir ou complementar conteúdos da matemática e computação que necessitam ser trabalhados em sala de aula.

Dessa maneira, propomos que as atividades de execução sejam aplicadas a alunos do 2º ano do ensino fundamental até o 3º ano do ensino médio; que as atividades de análise sejam aplicadas a partir do 6º ano do ensino fundamental ao 3º ano do ensino médio; mas as atividades de construção, atingirão resultados mais sofisticados, se forem executadas por alunos com um maior grau de maturidade, possivelmente, no ensino médio. Porém, o professor tem a liberdade de aplicar as atividades considerando o fato de os seus alunos terem ou não condições de executá-las e alcançar os objetivos desejados.

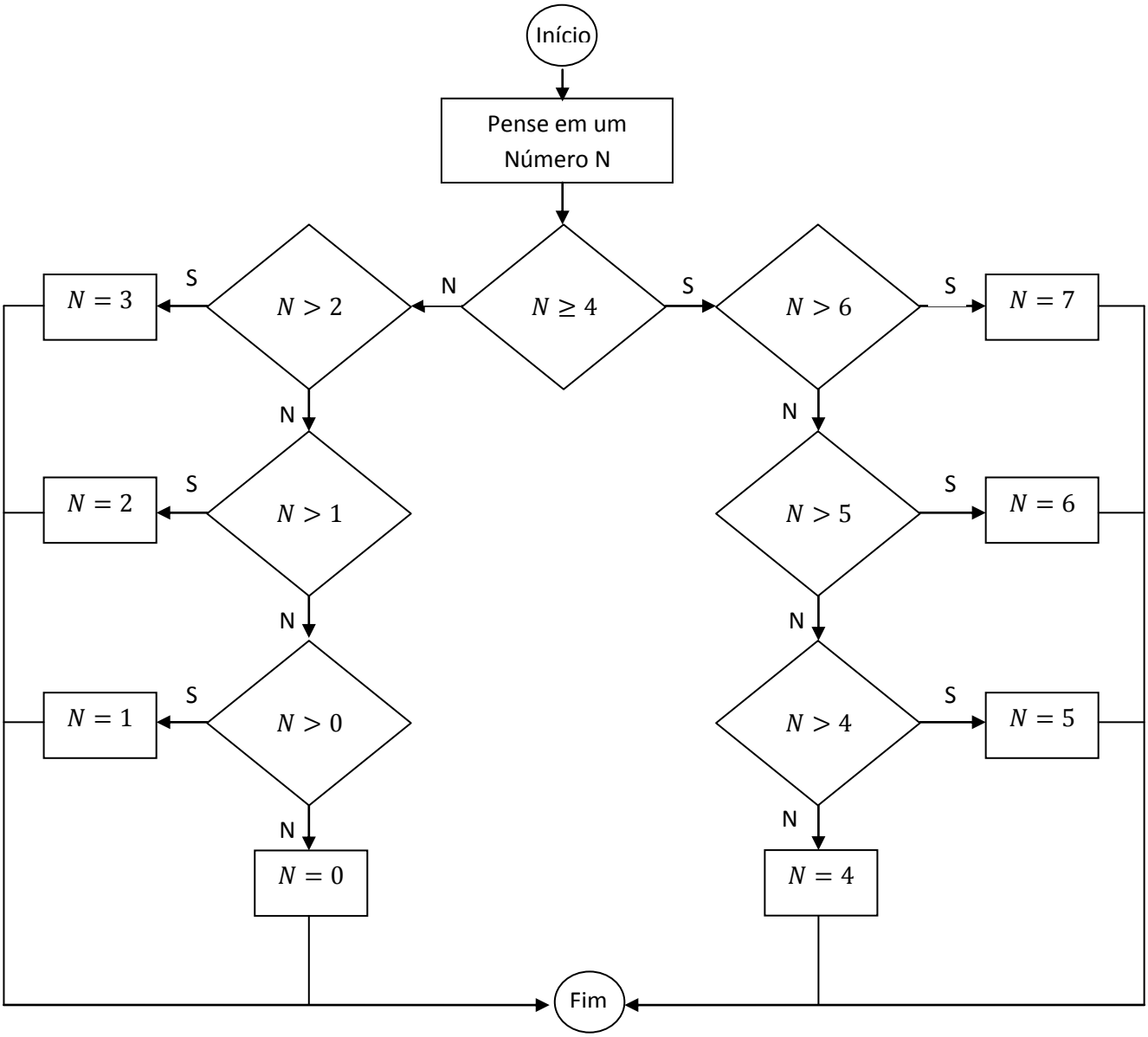
Atividade 1) Peça para uma criança pensar em um número de 0 a 7. Depois você pergunta: "O número é maior ou igual do que quatro?". Se sim, você segue o caminho do sim (no fluxograma 21); se não, você segue o caminho do não. Repita este processo até concluir qual foi o número que ele pensou.

Em seguida, peça para a criança ficar com a folha que contém o algoritmo de forma que somente ela possa enxergar. Pense em um número e peça para ela fazer as perguntas e concluir o resultado. Se o roteiro for seguido de forma correta a criança também determinará o número que você pensou.

O fluxograma 21 descreve um algoritmo que determina o número pensado. Este algoritmo pode ser facilmente adaptado para o caso em que o número pensado pertence a um intervalo de 0 a N, com $N \in \mathbb{N}^*$. Para complementar a atividade, o professor poderá solicitar que os alunos calculem o número máximo de perguntas necessárias para adivinhar o número pensado. Este valor é estimado em (N+1)/2 perguntas.

Note que para executar o algoritmo fluxograma 21 é suficiente que o estudante tenha a ideia de ordenação. Portanto, esta atividade pode ser aplicada a alunos do 2º ano do ensino fundamental e ser ampliada a alunos do ensino médio, desde que o professor aumente o intervalo o qual pertence o número pensado.

O objetivo dessa atividade é efetuar a busca pelo número pensado e possui a mesma ideia do algoritmo do fluxograma 20 de busca binária apresentado no capítulo 3. Não há duvidas de que estaremos introduzindo computação básica para esta criança.



Fluxograma 21 – Atividade 1

Atividade 2) Ao tratar da incomensurabilidade é interessante estabelecer com os alunos métodos para efetuar aproximações de números irracionais, como raízes quadradas não-exatas, o número pi (π), entre outros.

Matemáticos mesopotâmicos foram responsáveis por desenvolver processos algorítmicos eficientes, entre os quais está um para extrair a raiz quadrada. Trata-se de um

algoritmo simples, que ora é atribuído ao sábio grego Arquitas (428-365 a. C.), ora a Heron de Alexandria (100 d. C. aproximadamente), sendo até chamado de algoritmo de Newton.

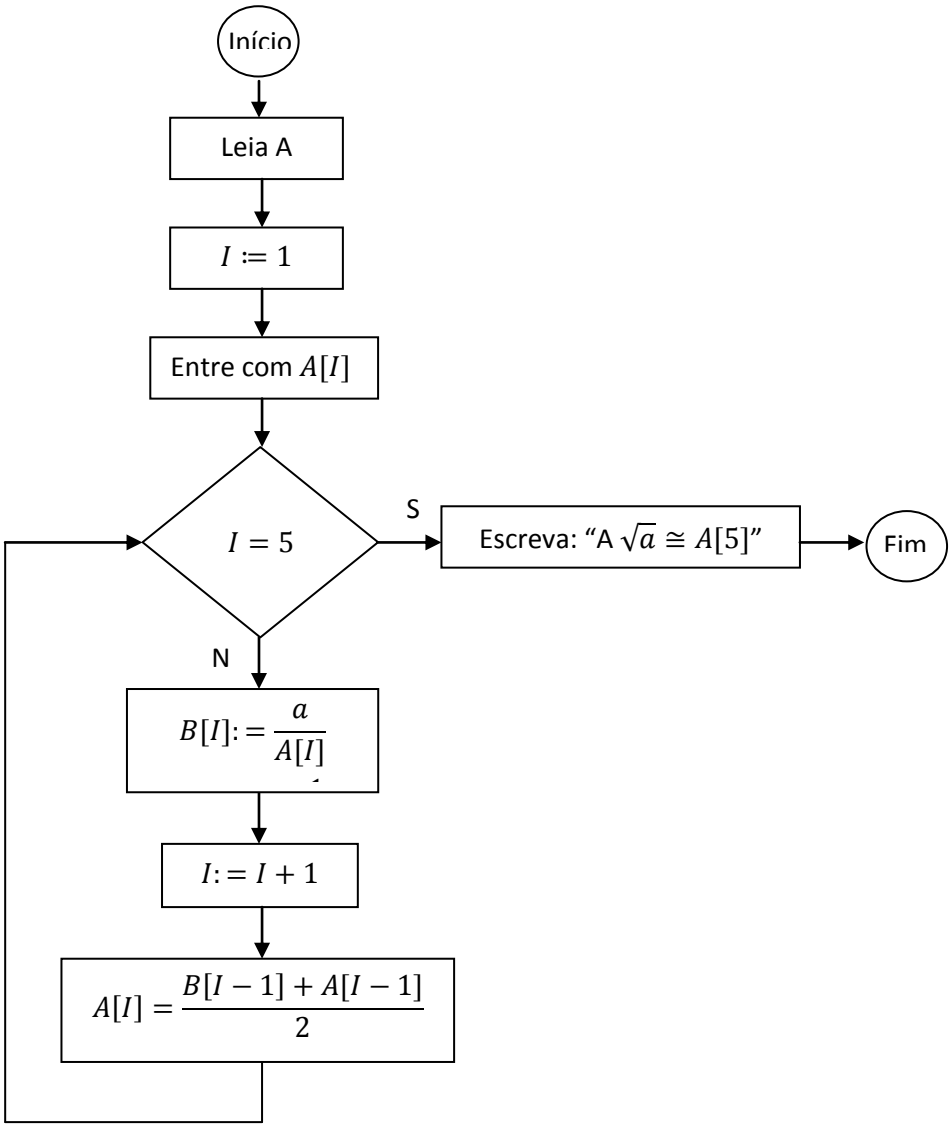
> [...] Seja $x = \sqrt{a}$ e a_1 uma primeira aproximação dessa raiz; b_1 uma segunda aproximação dada pela equação $b_1 = \frac{a}{a_1}$. Se a_1 é pequeno demais, b_1 é grande demais e vice-versa. Logo a média aritmética $a_2 = \frac{a_1 + b_1}{2}$ é uma nova aproximação plausível. Como a_2 é sempre grande demais, a seguinte, $b_2 = \frac{a}{a_2}$ será pequena demais e toma-se a média aritmética $a_3 = \frac{a_2 + b_2}{2}$ para obter um resultado ainda melhor; o processo pode ser continuado indefinidamente. [...] (BOYER, 1996, p. 19)

A partir do algoritmo proposto por mesopotâmicos para calcular raízes quadradas não exatas, construímos o fluxograma 22, o qual determina aproximações para raízes quadradas de não-quadrados perfeitos com 5 iterações. Esta rotina pode ser trabalhada em sala de aula com alunos do 8º ano do ensino fundamental ao 3º ano do ensino médio.

Inicialmente, o professor pode solicitar que os estudantes executem o algoritmo, com o objetivo de determinar a raiz quadrada de um número natural que não seja quadrado perfeito. Neste momento, ele pode ressaltar que os processos algorítmicos tem origem desde a Antiguidade e que a evolução do pensamento matemático é determinante no melhoramento do pensamento algorítmico e computacional.

Para construir o algoritmo representado pelo fluxograma 20 recorremos à ideia de sequências e também efetuamos buscas. O conceito de sequências recursivas pode ser introduzido. Alunos do 3º ano do ensino médio, já na fase de construção dos algoritmos, podem até construir o fluxograma 20 a partir das informações da citação anterior, além de executá-los.

Este algoritmo nos oferece uma boa aproximação para raízes quadradas de números que não são quadrados perfeitos e consiste, certamente, num ótimo exercício para complementar estudos sobre irracionalidade e sequências numéricas.



Fluxograma 22 – Atividade 2

As atividades 3 e 4 foram adaptadas do Banco de Questões (2012) e (2013) respectivamente, das Olimpíadas Brasileiras de Matemática das Escolas Públicas (OBMEP).

Atividade 3) (OBMEP – 2012) Começando com qualquer número natural não nulo é sempre possível formar uma sequência de números que terminam em 1, seguindo rapidamente as instruções abaixo:

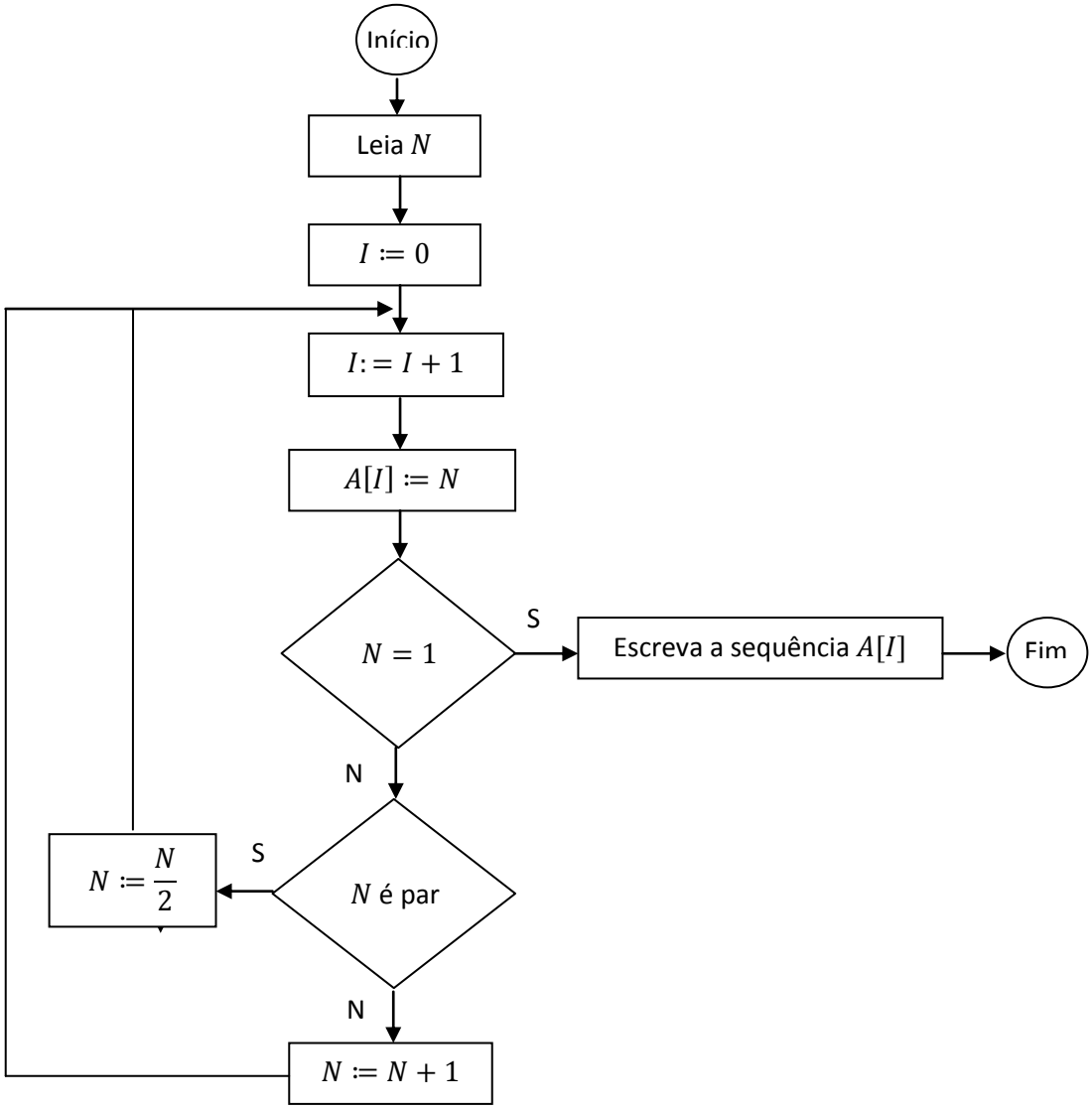
- se o número for ímpar, soma-se 1;
- se o número for par, divide-se por 2.

Por exemplo, começando com o número 21, forma-se a seguinte sequência:

$$21 \to 22 \to 11 \to 12 \to 6 \to 3 \to 4 \to 2 \to 1$$

Nessa sequência aparecem nove números; por isso, dizemos que ela tem comprimento 9. Além disso, como ela começa com um número ímpar, dizemos que ela é uma sequência ímpar.

O professor pode solicitar aos seus alunos que construam um algoritmo que determina uma sequência a partir das instruções anteriores, dado um número natural **n,** par ou ímpar. Neste momento esperamos que os alunos apliquem os conceitos abordados no capítulo 3, para construir o fluxograma ou pseudocódigo que descreverá o algoritmo requerido.



Fluxograma 23 – Atividade 3

Em seguida, podemos propor que eles escrevam uma sequência, utilizando o fluxograma 23, dado um número natural **n** qualquer (por exemplo $n = 37$) e respondam as seguintes questões:

a) Existem três sequências de comprimento 5, sendo duas pares e uma ímpar. Escreva essas sequências.

b) Quantas são as sequências pares e quantas são as sequências ímpares de comprimento 6? E de comprimento 7?

c) Existem ao todo 377 sequências de comprimento 5, sendo 233 pares e 144 ímpares. Quantas são as sequências de comprimento 16? Dessas, quantas são pares? Não esqueça de justificar sua resposta.

Atividade 4)(OBMEP – 2013) Na cidade de Autolândia, a numeração de placas de carros é feita através de números de três dígitos, portanto indo da placa 000 até a placa 999. Para diminuir a poluição, o prefeito Pietro decidiu implementar um rodízio de carros, estabelecendo os dias nos quais as pessoas podem usar os seus carros. As regras do rodízio são:

- Segunda-feira: somente carros com placa ímpar;
- Terça-feira: somente carros com placa cuja soma dos três dígitos é maior ou igual a 11;
- Quarta-feira: somente carros com placa cujo número é múltiplo de 3;
- Quinta-feira: somente carros com placa cuja soma dos três dígitos é menor ou igual a 14;
- Sexta-feira: somente carros com placa contendo pelo menos dois dígitos iguais;
- Sábado: somente carros cujo número na placa for estritamente menor do que 500;
- Domingo: somente carros cuja placa tenha os três dígitos menores ou iguais a 5.

Peça aos alunos para construírem um algoritmo para determinar os dias da semana que um determinado carro pode circular, a partir do número da sua respectiva placa.

Em seguida, o professor pode dividir a sala em sete grupos, um para cada dia da semana, e solicitar que cada grupo, construa um algoritmo para determinar se o número da placa do carro satisfaz o critério necessário para circular no seu respectivo dia da semana.

Esta atividade refere-se à atividade de construção e análise de algoritmos, que pode ser facilmente executada a partir dos algoritmos da matemática básica elaborados no capítulo 3. Neste momento, o professor poderá trabalhar critérios de divisibilidade, o algoritmo da divisão de Euclides e contagem.

Além disso, os algoritmos construídos podem auxiliar os alunos a responder às seguintes questões:

a. Em quais dias o carro com a placa 729 pode circular?
b. Maria, a esposa do prefeito, quer um carro que possa circular todos os dias exceto aos domingos. Qual placa ela deve ter?

c. O prefeito Pietro precisa de uma placa que o permita circular todos os dias. Que placa ele deve ter?
d. Por que todos os habitantes de Autolândia podem circular pelo menos uma vez por semana?

As atividades anteriores nos permitem abordar novos conteúdos matemáticos e revisar conhecimentos prévios em sala de aula, além de aprofundar os conceitos acerca de linguagem de programação. Observem que para construir os algoritmos anteriores utilizamos os fluxogramas, mas poderíamos tê-los feito, com o uso dos pseudocódigos, o que consiste num bom exercício para os estudantes. Acreditamos que uma introdução à computação através de noções básicas de linguagem de programação, pode ser feita durante as aulas de matemática e num primeiro momento, necessita da abordagem sobre os conceitos descritos no capitulo 2 e 3. Várias atividades podem ser desenvolvidas a partir das propostas que aqui fizemos, podendo ser elaboradas pelo próprio professor de Matemática.

5. CONSIDERAÇÕES FINAIS

A sociedade atual preocupa-se com o desenvolvimento das Ciências e Tecnologias e o considera crucial para o seu progresso. Sabemos que muitas estratégias têm sido aplicadas para que haja o tratamento adequado da informação. Entretanto, sentimos a necessidade de apresentar, em nível básico, conteúdos da aritmética relativos à Ciência da Computação, pois constatamos o aumento da demanda de profissionais dessa área caracterizado pela carência de competências e habilidades que a maioria dos estudantes apresenta com a Matemática e áreas afins.

Dentro dessa perspectiva, propomos o ensino-aprendizado de conceitos básicos de linguagem de programação baseados na construção e análise de algoritmos no ensino básico. Esclarecemos que dispositivos como esquemas estruturados (pseudocódigos) e fluxogramas apresentados neste trabalhado representam processos em geral, mesmo os não 'matemáticos'. E que ao contrário do que muitas pessoas podem pensar, o princípio de funcionamento do computador é bem simples e pode ser trabalhado desde o ensino fundamental, bastando, que se introduza a ideia geral do funcionamento de uma máquina que pode executar algumas tarefas básicas e repetitivas.

Acreditamos que esta metodologia contribui para a formação do pensamento computacional a qual abrange a identificação de tarefas cognitivas que podem ser feitas de forma mais rápida e eficiente por um computador. Além disso, aprender conceitos básicos de programação por meio de algoritmos possibilita um complemento ao desenvolvimento de habilidades e competências aplicadas à resolução de problemas, já que criar algoritmos requer um alto grau de criatividade, raciocínio lógico, objetividade e clareza.

Sabemos que são muitas as dificuldades que estão por vir, pois para que a inserção do ensino de Ciência da Computação no ensino básico ocorra com sucesso, faz-se necessário a preparação adequada de profissionais da educação. Porém, este trabalho pode ser desenvolvido por professores de matemática, desde que sua formação contemple, pelo menos, a introdução dos conceitos abordados. Com isso, elaboramos algumas atividades que podem ser executadas nas aulas de matemática, ou como propõe a Sociedade Brasileira de Computação (SBC), pela inserção da Introdução à Ciência da Computação como uma disciplina do ensino básico. São atividades que propiciam, ao longo de sua resolução, o

desenvolvimento de novos conceitos e a revisão de conhecimentos prévios tanto da matemática quanto da computação.

6. REFERÊNCIAS BIBLIOGRÁFICAS

BELTRÁN, J; FARFÁN, J; HILÁRIO, M; FRANCO, T. **OBMEP-Banco de Questões 2013.** Rio de Janeiro, RJ. IMPA, 2013.

BROOKSHEAR, J. G. **Ciência da Computação Uma Visão Abrangente**. 7ª Edição. São Paulo: Bookman, 2003.

BLIKSTEIN, P. **O pensamento Computacional e a Reinvenção do Computador na Educação**. Disponível em: http://cgceducacao.com.br/canal.php?c=4&a=10552. Acesso em Julho de 2013.

HEFEZ, A. **Elementos de Aritmética**. 2ª edição. Rio de Janeiro: SBM, 2011.

INSTITUTO NACIONAL DE MATEMÁTICA PURA E APLICADA. **OBMEP - Banco de questões 2012**. Rio de Janeiro,RJ. IMPA. 2012.

FARREL, J. **Lógica e Design de Programação**. Tradução André Schifnagel Avrichir. 5ª Edição. São Paulo: Cengage Learnig, 2010.

LEAL, W. S. **O Ensino de Algoritmos no Ensino Medio: Por que Não**. 2009. Disponível em: www.unigranrio.br/.../mestrado/ensino.../dissertacao_willian_silva.pdf.

LEAL, W. S.; LOZANO, A. R. G. **A Viabilidade do Ensino de Algoritmos no Ensino Médio**. 2009. Disponível em: www.pucpr/eventos/eduserie/eduserie2009/anais/.../3204_1691.pdf. Acesso em Janeiro 2013.

LIMA, A. C. **Lógica Formal: Origens e Aplicações**. Salvador: Quarteto, 2010.

NUNES, J. D. **Ciência da Computação na Educação Básica**. Disponível em: www.jornaldaciencia.org.br/Detalhe.jsp?id=79207. Acesso em Julho de 2013.

ONUCHIC, L. R.; ALLEVATO, N. S. G. **Pesquisa em Resolução de Problemas: caminhos, avanços e novas perspectivas**. Disponível em: www.redalyc.org/articulo.oa?id=291223514005. Acesso em Junho de 2013.

SECRETÁRIA DE EDUCAÇÃO BÁSICA. **Ciências da Natureza, Matemática e Suas Tecnologias. Orientações Curriculares para o Ensino Médio**, volume 2. Brasília: Ministério da Educação, 2008.

SIEBRA, S. A. ; SILVA, D. R. D. **Prática de Ensino de Algoritmos**. Recife. Universidade Federal Rural de Pernambuco, 2009.

TAUB, H. **Circuitos Digitais e Microprocessadores**. São Paulo: Mc Graw – Hill do Brasil, 1984.

TENÓRIO, R. M. **Computadores de Papel: Maquinas Abstratas para o Ensino Concreto**. São Paulo: Cortez, Autores Associados, 1991.

TERADA, R.; SETZER, W. **Introdução à Computação e à Construção de Algoritmos**. São Paulo: Makron Books, 1992.

VALENTE, J. A. **Computadores e Conhecimento: representando a educação**. Campinas – SP. Gráfica Central da UNICAMP, 1993.

WIKIPÉDIA. **Tese de Church-Turing**. Disponível em: www.pt.wikipedia.org/wiki/Tese_de_Church-Turing. Acesso em Julho de 2013.