Introdução

Objetivos do Capítulo

Os objetivos principais deste Capítulo são mostrar alguns aspectos da história da computação para o aluno que inicia o curso de Computação e definir, ainda que informalmente, alguns termos e palavras chaves que os profissionais da área de computação usam.

Sucessos e Fracassos da Computação

A história do desenvolvimento dos computadores tem sido impressionante. O avanço da tecnologia e a penetração da computação na nossa vida é inegável. Embora a história deste fantástico desenvolvimento seja recente e esteja bem documentada, há lacunas e controvésias impressionantes sobre diversos pontos. Iremos ver histórias de espionagem e brigas na justiça por roubo de idéias. Há oportunidades perdidas e gente que soube aproveitar a sua chance. Há verdades estabelecidas que devem ser revistas.

O avanço na tecnologia dos computadores se deu em passos tão largos que os primeiros computadores parecem estar tão distantes no tempo quanto a pré-história. O aumento de velocidade, desde os anos 40, foi da ordem de 100000, enquanto que o custo dos computadores caiu de milhões de dólares para valores em torno poucos milhares. As primeiras máquinas tinham milhares de válvulas, ocupavam áreas enormes e consumiam kilowatts de energia. O microprocessador Pentium, lançado em 1993, tinha em torno de 3,1 milhões de transistores, ocupava uma área de aproximadamente 25 cm2 e consumia alguns watts de energia, custando da ordem de 1000 dólares, somente o microprocessador.

No entanto, esta história de redução de tamanho, aumento de velocidade e diminuição de gasto de potência já tem uma data para terminar. Em 1965, Gordon Moore, um dos fundadores da Intel, um dos maiores fabricantes de circuitos integrados do mundo enunciou o que ficou conhecido como a Lei de Moore. "Cada novo circuito integrado terá o dobro de transistores do anterior e será lançado em um intervalo de 18 ou 24 meses." Moore achava que esta lei seria válida somente até 1975, no entanto vem sendo válida até hoje. Para ver como ele estava certo, basta ver, na tabela abaixo, a

evolução dos microprocessadores usados em nossos computadores.

Ano	Processador	Transistores
1971	4004	2.250
1972	8008	2.500
1974	8080	5.000
1982	80286	120.000
1985	80386	275.500
1989	80486 DX	1.180.000
1993	Pentium	3.100.000
1997	Pentium II	7.500.000
1999	Pentium III	24.000.000
2000	Pentium 4	42.000.000

Rapidamente os transistores, que são os tijolos de que são compostos os circuitos eletrônicos, estão diminuindo de tamanho e estamos nos aproximando da fronteira final, os eletrons. Devemos nos lembrar que toda a tecnologia atual está baseada em fluxo de eletrons, a corrente elétrica. Os fios conduzem correntes de eletrons, então se formos além disto estaremos em outro domínio. Já se fala em empregar o sentido de rotação dos eletrons para indicar os zeros e uns.

No entanto, promessas não foram cumpridas e falhas gigantescas aconteceram. Como em diversas questões os artistas apontam o que não conseguimos ou queremos ver e mostram que o rei está nu. Há uma frase de Picasso em que ele diz "Computadores são estúpidos, eles somente conseguem responder perguntas". Esta frase expõe com ironia um fracasso da comunidade de computação que havia prometido criar rapidamente computadores inteligentes, computadores que poderiam se questionar e nos questionar. Muitos acreditaram nesta promessa como mostra o filme "2001 - Uma Odisséia no Espaço" de Stanley Kubrik que estreou em 1968. O filme foi baseado no conto "The Sentinel", escrito em 1950 por Arthur Clark, mestre da ficção científica. Neste filme o enlouquecido computador HAL 9000, que era capaz de ver, falar, raciocinar, etc, mata quase todos os tripulantes de uma nave espacial. Ora, estamos em 2001 e não estamos nem perto de ver um computador como o HAL. Gostaria de ver algo parecido, poderia até ser um computador louco de pedra como HAL. A imagem abaixo, que ficou famosa, mostra o que seria o um dos olhos de HAL

Outra imagem de sucesso fantástico e a Internet. Frequentemente ouvimos que a Internet foi o meio de comunicação que mais rapidamente se difundiu pelo mundo. Pode parecer verdade, já que conhecemos tantos internautas e as empresas estão se atropelando para fazer parte desta onda. Embora esta corrida provocou alguns acidentes e sonhos de riqueza se esfumaçaram no ar. Hoje pode-se fazer quase tudo pela Internet, namorar, comprar, pagar contas, fazer amigos, estudar, jogar, etc. Quem sabe, em um futuro próximo, voltaremos à Grécia Antiga e nos reuniremos em uma enorme praça virtual para democraticamente discutir nossas leis, dispensando intermediários, que em alguns países somente atrapalham. No entanto, em um país como o Brasil, há tanta gente que não tem nem acesso a um telefone, quanto mais à Internet. Então qual é a verdade, estamos conectados ou não? Os números servem tanto para mostrar como ocultar a realidade, ou quem sabe a verdade tem muitas faces. É verdade que a Internet se expandiu vertiginosamente, quando levamos em conta o número de anos que os meios de comunicação levaram para atingir 50 milhões

de usuários. A Tabela abaixo mostra estes dados.

Meio	Anos para atingir 50 milhões de consumidores
Telefone	70
Rádio	38
Televisão	13
Internet	5

É conhecida a história dos dois homens miseráveis que tinham um frango assado para matar a fome. Os manipuladores dizem que estatisticamente cada um comeu meio frango e podemos ir dormir todos satisfeitos com nossa consciência tranquila. Faltou dizer que um comeu o frango todo. Nesta história da Internet faltou levar em conta a população mundial e aí podemos continuar impressionados, mas nem tanto. A Tabela abaixo mostra dados sobre a população mundial e a presença média de cada um destes meios de comunicação na população.

Sistema	Década de lançamento	Atingiu 50 milhões	População Mundial	Um sistema para cada
Telefone	1900	1970	3,8 bilhões	76 pessoas
Rádio	1930	1968	3,7 bilhões	74 pessoas
Televisão	1950	1964	3,2 bilhões	64 pessoas
Internet	1990	1995	5,8 bilhões	116 pessoas

Os dados da tabela mostram que a Internet teve um crescimento rápido mas não está tão disseminada quanto os outros meios de comunicação. Há um enorme contingente de seres humanos que não dispõem de meios e conhecimentos para se aproximar desta praça virtual e portanto não poderão discutir sobre o seu futuro ou pelo menos dar uma volta para ver o que está acontecendo. Diferentemente dos outros meios, a Internet, ainda requer que seus usuários possuam mais informações e treinamento do que os outros meios, colocando, portanto, uma barreira inicial mais alta.

Um Pouco de História

A primeira tentativa de se criar uma máquina de contar foi o ábaco. A palavra vem do árabe e significa pó. Os primeiros ábacos eram bandejas de areia sobre as quais se faziam figuras para representar as operações. Aparentemente os chineses foram os inventores do ábaco de calcular. No entanto, há controvérsias, e os japoneses também reivindicam esta invenção, que eles chamam de soroban. Além disto há os russos, que inventaram um tipo mais simples chamado de tschoty. São conhecidos exemplares de ábaco datando de 2500 A.C. A figura ilustra um exemplar com as suas contas e varetas.

O primeiro instrumento moderno de calcular do tipo mecânico foi construído pelo filósofo, matemático e físico francês Blaise Pascal, em 1642, na cidade de Rouen, quando ele tinha 19 anos. Pascal desenvolveu uma máquina de calcular, para auxiliar o seu trabalho de contabilidade. A engenhoca era baseada em 2 conjuntos de discos interligados por meios de engrenagens: um para a introdução dos dados e outro que armazenava os resultados. A máquina utilizava o sistema decimal para os seus cálculos de maneira que quando um disco ultrapassava o valor 9, retornava ao 0 e

aumentava uma unidade no disco imediatamente superior.

Pascal recebeu uma patente do rei da França para que lançasse sua máquina no comércio. A comercialização de suas calculadoras não foi satisfatória devido a seu funcionamento pouco confiável, apesar de Pascal ter construido cerca de 50 versões. As máquinas de calcular, descendentes da Pascalina, ainda hoje podem ser encontradas em uso por algumas lojas de departamentos. Antes de morrer, aos 39 anos, em 1662, Pascal que deu enormes contribuições em vários campos da ciência, ainda teve tempo de criar uma derivação de sua calculadora, a caixa registradora.

Em 1666, Samuel Morland adaptou a calculadora de Pascal para que resolvesse multiplicações por uma série de somas sucessivas. Independentemente, em 1671 Leibniz projetou uma calculadora que podia somar e multiplicar. Esta calculadora foi terminada em 1694.

O primeiro computador de uso específico foi projetado em 1812, portanto há mais de 180 anos atrás, por Charles Babbage (1791-1871), que batizou sua máquina de *Difference Engine*. A motivação de Babbage era resolver polinômios pelo método das diferenças. Naquele tempo as tábuas astronômicas e outras tabelas eram calculadas por humanos, em métodos tediosos e repetitivos.

Difference Engine

Entre 1820 e 1822, ele construiu uma outra máquina mais avançada e capaz de calcular polinômios de segunda ordem com até 6 dígitos de precisão. Em seguida tentou construir outra capaz de calcular polinômios de até sexta ordem com 26 dígitos de precisão. Este esforço durou até 1833. Esta máquina, inteiramente mecânica tinha as seguintes características:

- Arrendondamento automático;
- Precisão dupla;
- Alarmes para avisar fim de cálculo;
- Impressão automática de resultados em placas de cobre.

Este projeto não foi completado e gastou dinheiro do governo inglês e possivelmente a maior parte da fortuna pessoal de Babbage. Durante os anos 1830 tentou construir outra de uso mais geral com as seguintes características:

- 50 dígitos decimais de precisão;
- Memória para 1000 destes números (165000 bits);
- Controle por meio de cartões perfurados das operações e endereços dos dados;
- Tempo de soma e subtração igual a 1 segundo; tempo de multiplicação e divisão igual a 1 minuto;
- Subrotinas;
- Arrendondamento automático e deteção de transbordo (overflow);

Babagge nunca conseguiu terminar este ambicioso projeto. No entanto, os mais importantes conceitos de computação, que foram somente redescobertos nos anos 40 do século vinte, foram pensados por Charles Babbage.

Um fato curioso é que entre os auxiliares de Babagge estava Ada Lovelace, que é considerada a primeira programadora de computadores. Este título não agrada a muita gente que diz que ela não era realmente uma programadora. Ada Lovelace estudava Matemática com De Morgan, que provou um dos teoremas básicos da álgebra Booleana, que é a base matemática sobre a qual foram desenvolvidos os projetos dos modernos computadores. No entanto, não havia nenhuma ligação entre o trabalho do projetista dos primeiros computadores e o do matemático que estudava o que viria a ser o fundamento teórico de todo a computação que conhecemos hoje.

Os computadores da era moderna (antes e durante a segunda guerra) foram construídos com relés e válvulas. Quatro grupos ou indivíduos foram pioneiros nestes anos e os três últimos pesquisadores construíram computadores com relés.

1. **John Atanasoff:** depois de um caso judicial, passou a ser considerado o construtor do primeiro computador digital (1939, Iowa State University)
2. **Howard Aiken:** (1937-1944, Harvard University)
3. **George R. Stibitz:** (1938-1940, Bell Telephone Labs) Primeiro a usar um computador remotamente.
4. **Konrad Zuze:** (1936-1940, Berlin Technishe Hochsule)
5. J **P Eckert e J Mauchly:** (1946, Universidade da Pensilvânia) Primeiro computador digital operacional chamado de ENIAC (Electronic Numerical Integrator and Calculator).

Durante muitos anos o computador ENIAC foi considerado o primeiro computador construído. A máquina projetada pelos Drs. Eckert and Mauchly era uma monstruosidade. Quando foi terminada, o ENIAC enchia um laboratório inteiro, pesava trinta toneladas, e consumia duzentos quilowatts de potência.
Ele gerava tanto calor que teve de ser instalado em um dos poucos espaços da Universidade que possuia sistemas de refrigeração forçada. Mais de 19000 válvulas, eram os elementos principais dos circuitos do computador. Ele também tinha quinze mil relés e centenas de milhares de resistores, capacitores e indutores. Toda esta parafernália eletrônica foi montada em quarenta e dois painéis com mais 2,70 metros de altura, 60 cm de largura e 30 de comprimento. Eles foram montados na forma da letra U. Uma leitora de cartões perfurados (a maioria dos alunos nunca ouviu falar disto!) e uma perfuradora de cartões eram usados para entrada e saída de dados. Acreditem, mas houve uma época em que computadores eram programados com cartões de papel perfurados. Nada de teclados com terminais de vídeos ou estas coisas que hoje são lugar comum em todos computadores.

Como um dado interessante, foi divulgado recentemente que um computador chamado COLOSSUS entrou em operação secretamente na Inglaterra em 1943. Este fato e a batalha judicial pelo reconhecimento de quem construiu o primeiro computador digital mostram que, apesar de sua história ter se iniciado recentemente, o passado do computador não é tão claro como poderia parecer.

Os tempos de execução do ENIAC são mostrados na tabela a seguir. Compare estes tempos com os

tempos dos computadores atuais que estão na ordem de nano segundos, ou 10^{-9} segundos.

Operação	Tempo
soma	200 microsegundos
multiplicação	2,8 ms
divisão	6,0 ms

O primeiro computador a usar o conceito de programa armazenado em memória a ser terminado foi o EDSAC, (Electronic Delay Storage Automatic Calculator) em 1949, na Universidade de Cambridge, Inglaterra. O conceito de programa armazenado foi fundamental para o progresso da computação. Os primeiros computadores eram programados por fios que os cientistas usavam para conectar as diversas partes. Uma vez que um programa terminava, estes cientistas trocavam os fios de posição de acordo com a nova tarefa a ser executada. Com o programa armazenado na memória juntamente com os dados, não era mais necessário interromper as atividades. Carregava-se o programa na memória, uma tarefa extremamente rápida, junto com os dados e dava-se partida no programa. Ao término da execução do programa passava-se imediatamente para a próxima tarefa sem interrupções para troca de fios.

Costumava-se dividir os projetos de computadores em gerações. Hoje em dia como a taxa de evolução é muito grande não se usa mais este tipo de terminologia. No entanto é interessante mencionar estas divisões.

- **Primeira Geração:** Os computadores construídos com relés e válvulas são os da primeira geração. Estes computadores consomem muita energia e espaço.
- **Segunda Geração:** Os computadores da segunda geração foram construídos com transistores que tinham a vantagem de serem mais compactos e consumirem muito menos energia. Por gerarem menos calor eram máquinas mais confiáveis.
- **Terceira Geração:** Com o advento dos circuitos integrados, que são componentes em que vários transistores são construídos em uma mesma base de silício, chegamos aos computadores de terceira geração. Com a integração o tamanho dos computadores e seu consumo diminuiu ainda mais e aumentou a capacidade de processamento.
- **Quarta Geração:** Os computadores de quarta geração utilizavam circuitos com a tecnologia VLSI (Very Large Scale Integration).

Este processo de miniaturização continuou e por isto hoje não se usa mais estes termos devido a dificuldade de caracterização das gerações e da velocidade com que novos lançamentos são feitos. A ultima vez que se tentou usar este termo fazia-se referência à computadores de quinta geração. Neste caso a referêencia não estava relacionada com a tecnologia de fabricação, mas à forma como os usuários iriam se comunicar com as máquinas. Os cientistas estavam procurando criar computadores que se comunicariam por meio de voz, visão, etc. O sucesso neste caso não foi tão grande e ainda estamos no início desta escalada.

O *Hardware*

Um típico diagrama em blocos de um computador digital monoprocessado esta mostrado na Figura a seguir.

Na linguagem mais técnica o que normalmente conhecemos por **processador** é chamado de **Unidade Central de Processamento** (UCP), em inglês Central Processing Unit (CPU). Uma UCP integrada em um único circuito é chamada de **microprocessador**. Os microprocessadores atuais já incluem circuitos que normalmente ficavam fora da UCP, tais como processadores de ponto flutuante, que são unidades especiais que fazem diretamente operações aritméticas com valores reais. Um microprocessador com memória e alguns periféricos forma o que se chama de **microcomputador**. Os modulos que constituem a UCP são os seguintes:

- **Unidade de Controle:** comanda a operação do computador. Esta unidade lê da memória tanto as instruções como os dados e comanda todos os circuitos para executar a instrução lida da memória. Atualmente as unidades de controle são capazes de executar mais de uma instrução por ciclo de máquina. A técnica mais comum para conseguir este paralelismo é conhecida como *pipelining*, que será detalhada mais adiante.
- **Unidade Aritmética:** local onde as transformações sobre os dados acontecem. Somente na unidade aritmética ocorrem transformações nos dados. Atualmente esta unidade é bastante sofisticada e diversos métodos são empregadas para acelerar a execução das instruções. Alguns processadores duplicam circuitos para permitir que mais de uma operacão aritmética seja executada em um determinado instante. É muito comum a existência de uma unidade aritmética para executar instruções que operam sobre números inteiros e outra sobre números reais.
- **Unidade de Entrada e Saída:** controla a comunicação com os usuários do computador e os equipamentos periféricos tais como discos e impressoras. Em alguns computadores esta unidade não existe independentemente, sendo distribuída pela Unidade Central de Processamento.

O termo *pipelining* que mencionamos acima se refere ao modo como o processador paraleliza a execução de instruções. Este termo pode ser traduzido por linha de montagem, porque é exatamente isto que a técnica faz, uma linha de montagem de instruções. Por exemplo, em uma linha de montagem de uma fábrica de automóveis, mais de um automóvel é montado ao mesmo tempo. No início da linha o carro não existe. Em cada estágio da linha de montagem os operários vão adicionando partes e ao fim da linha sai um carro novo. Se você olhar a linha poderá ver carros em diversos estágios da montagem. Repare que a linha não pára, e a montagem de um carro não espera que o que começou a ser montado antes dele esteja terminado. Nas linhas de montagem muitos carros são montados ao mesmo tempo. O efeito final é que cada carro leva bastante tempo para ser montado mas como vários vão sendo montados ao mesmo tempo, alguém que se colocasse no final da linha de montagem teria a impressão que cada carro leva muito pouco tempo para ser montado. O mesmo acontesse com a linha de montagem de instruções.

Os dados dentro do computador podem ser armazenados ou em registradores (ficam internamente ao processador), na memória ou em periféricos (discos, fitas, etc). Os dados e instruções na memória são apontados por **endereços**. Ou seja, dado um endereço a memória devolve o seu conteúdo. Isto é similar ao o que ocorre quando enviamos uma carta, o endereço faz com que o carteiro saiba onde ele deve entregar a correspondência. Em operação normal, toda vez que o processador precisa de um dado ele envia um pedido de leitura ou escrita à memória junto com o

endereço da memória onde o dado está.

A memória do computador é composta de **bits**, a menor unidade de informação que o computador armazena. Um bit pode conter o valor 0 ou 1, que são os dígitos usados na base dois, a base usada nos computadores. Um conjunto de 8 bits forma o **byte**. Uma **palavra de memória** é um conjunto de bytes. Atualmente a maioria dos computadores tem palavras de memória com largura de 32 (4 bytes) ou 64 (8 bytes) bits. Observar que estamos falando de memória e não do tamanho da palavra que o computador pode processar. Um computador pode ter capacidade de processar 64 bits de cada vez. Caso sua memória tenha palavras de 32 bits o processador deverá ler duas palavras da memória para pode executar suas instruções.

Devido a base 2 o fator kilo tem um significado diferente em computação. Por exemplo 1 Kbyte de memória corresponde a 2 elevado a 10 (2^{10}), ou seja 1024 bytes. Da mesma forma 1 Megabyte corresponde a 1024 x 1024 bytes e 1 Gigabyte é igual a 1024 x 1024 x 1024.

Quando se fala de memória duas siglas vêm logo ao nosso pensamento: ROM e RAM. Estas siglas têm diversas variações (PROM, EPROM, DRAM, etc), mas os princípios básicos são os mesmos. Estas siglas indicam dois tipos de memória que são muito usadas em computadores. A sigla básica ROM significa Read Only Memory, ou seja memória de somente de leitura e RAM (Random Access Memory) significa memória de acesso randômico, ou seja memória que se pode ler em qualquer endereço.

A sigla RAM é muito confusa porque em uma memória ROM também se pode ler em qualquer endereço. A diferença real é que na RAM se pode ler e escrever com a mesma velocidade e na ROM somente se pode ler os dados, a escrita é uma história mais complicada. A ROM normalmente contém dados que não podem ser modificados durante o funcionamento do computador. Outro tipo de dados armazenados em ROMs são os que não devem ser perdidos quando o computador é desligado. Exemplos de uso de ROM são as memórias que armazenam os programas que são executados quando os computadores são ligados. Um computador ao ser ligado deve ter um programa mínimo capaz de iniciar o funcionamento normal, caso contrário seria como uma pessoa que perdeu totalmente a memória.

As primeiras memórias deste tipo eram gravadas nas fábricas e nunca mais eram modificadas. Isto trazia algumas dificuldades, por exemplo, quando um programa era atualizado. Para resolver este tipo de problemas surgiram as PROMs, que são ROMs programáveis. Ou seja é possível desgravar o conteúdo antigo e gravar novos programas nesta memória. Antigamente este era um processo complicado e exigia que a memória fosse retirada fisicamente do circuito e colocada em dispositivos especiais capazes de apagar o conteúdo antigo. Em seguida um circuito programador de PROMs era usado para gravar o novo conteúdo e somente após tudo isto a memória era recolocada no local. O computador ficava literalmente sem a memória dos programas iniciais. Hoje em dia existm PROMs que podem ser apagadas e regravadas muito facilmente. Por exemplo, as EEPROMs (Eletricaly Erasable PROM), que são memórias que podem ser apagadas eletricamente sem a necessidade de serem retiradas dos circuitos.

As memórias RAMs são as memórias onde os nossos programas comuns rodam. Elas são modificáveis e de acesso rápido tanto na leitura quanto na gravação. Muitas siglas aparecem e desaparecem quando falamos de memórias RAM. Existem as DRAM, memórias EDO, SIMM, etc.

Tudo isto ou se refere ao método de acesso dos dados na memória ou a tecnologia de construção ou a outra caracterítica acessória. O certo é que todas elas tem como caracterítica básica o fato de os acessos de leitura e escrita podem ser feitos na mesma velocidade.

Como já mencionamos os dados não ficam guardados somente na memória, há também os periféricos. Há periféricos de entrada, outros de saída e alguns que servem tanto para entrada como saída de dados. Periféricos não servem somente para armazenar dados. Há periféricos que são usados para permitir a interação entre os usuários e o computador. A tabela a seguir ilustra alguns destes periféricos.

Entrada	**Saída**	**Ambos**
Teclados	Impressoras	Discos Rígidos
Mouse	Vídeo	Fitas Magnéticas
CD-ROM	Plotter	Disquetes
Scanner		

O *Software*

Tudo isto que sobre o que acabamos de escrever constitui o *hardware* do computador, o que se vê e o que se toca. A partir de agora falaremos no *software*, o não se vê nem se toca, mas também está lá.

Para que um computador execute alguma tarefa primeiro se desenvolve um algoritmo, que é a receita de como o problema deve ser resolvido. Em seguida este algoritmo deve ser traduzido para uma linguagem que possa ser entendida pelo computador, o programa. No início da computação os programas eram escritos diretamente em linguagem de máquina (assembly), que é a linguagem que o computador *entende*. Este tipo de programação pode levar a se escrever programas muito eficientes, devido ao controle quase que total do programador sobre a máquina. No entanto devido ao fato de ser uma linguagem próxima do computador e afastada da maneira de raciocionar do ser humano é mais dificil de ser usada. Além deste fato há outros problemas como dificuldade de leitura e manutenção dos programas, maior tempo de desenvolvimento, etc. Para evitar estes problemas foram desenvolvidas as linguagens de programação de alto nível. Algins exemplos de linguagens de programação são:

- **Fortran:** Usada em programação científica e engenharia;
- **Pascal:** Usada em ensino de linguagens e desenvolvimento de sistemas;
- **COBOL:** Usada em ambientes comerciais;
- **Basic:** O nome diz tudo;
- **C:** Mesmas características do Pascal com facilidades que permitem mais facilmente manipulação dos bits;
- **C++:** Linguagem originária do C com metodologia de orientação à objetos;
- **Delphi:** Linguagem originária do Pascal com metodologia de orientação à objetos;
- **Lisp e Prolog**

  : Linguagens muito usadas para desenvolver programas de Inteligência Artificial;

Hoje em dia a maior parte dos usuários de computadores não são programadores e sim

pessoas que usam programas para resolver seus problemas do dia a dia. Estes programas são muitas vezes chamados de aplicativos. Aplicativos típicos que rodam nos computadores são: editores de texto, planilhas eletrônicas, compiladores, bancos de dados, jogos, etc.

Para gerenciar todos os recursos do computador existe um programa especial normalmente chamado de Sistema Operacional (S. O.). O Sistema Operacional controla a alocação de recursos tais como: comunicação com os usuários, espaço em discos, uso de memória, tempo que cada programa pode rodar, etc. Sistemas operacionais conhecidos são os baseados no padrão UNIX, por exemplo o LINUX, o DOS que é um sistema que os PCs usavam, e os da família Windows.

Exercícios

1. **Quais as diferenças entre um microprocessador e o microcomputador?**
2. **Dê exemplos de microprocessadores e de microcomputadores.**
3. **O que**
4. **Qual o número exato de bytes em 64Kbytes?**
5. **Se você já usa computadores, liste alguns aplicativos que você normalmente usa.**
6. **Defina brevemente Sistema Operacional.**
7. **Qual a diferença básica entre memórias ROM e RAM?**
8. **Qual a base que o computador usa para representar os números?**
9. **Faça três listas, uma de periféricos de entrada, outra de periféricos de saída e finalmente uma de periféricos de entrada e saída.**

Data da última atualização: 31/08/2001

O que são Algoritmos?

Objetivos do Capítulo

Os objetivos deste capítulo são apresentar o que são algoritmos, quais são as formas de representá-los e algumas das técnicas usadas durante sua criação e desenvolvimento.

Introdução

Para resolver um problema no computador é necessário que seja primeiramente encontrada uma maneira de descrever este problema de uma forma clara e precisa. É preciso que encontremos uma seqüência de passos que permitam que o problema possa ser resolvido de maneira automática e repetitiva. Esta seqüência de passos é chamada de algoritmo. Um exemplo simples e prosaico de como um problema pode ser resolvido se fornecermos uma seqüência de passos que mostrem a solução é uma receita de bolo.

A noção de algoritmo é central para toda a computação. A criação de algoritmos para resolver os problemas é uma das maiores dificuldades dos iniciantes em programação em computadores.

No seu livro Fundamental Algorithms vol. 1 Donald Knuth apresenta uma versão para a origem desta palavra. Ela seria derivada do nome de um famoso matemático persa chamado Abu Ja´far Maomé ibn Mûsâ al-Khowârism (825) que traduzido literalmente quer dizer Pai de Ja´far, Maomé, filho de Moisés, de Khowârizm. Khowârizm é hoje a cidade de Khiva, na ex União Soviética. Este autor escreveu um livro chamado *Kitab al jabr w´al-muqabala* (Regras de Restauração e Redução). O título do livro deu origem também a palavra Álgebra.

O significado da palavra é muito similar ao de uma receita, procedimento, técnica, rotina. **Um algoritmo é um conjunto finito de regras que fornece uma sequência de operações para resolver um problema específico**.

Um algoritmo opera sobre um conjunto de entradas (no caso do bolo, farinha ovos, fermento, etc.) de modo a gerar uma saída que seja útil (ou agradável) para o usuário (o bolo pronto). Um algoritmo tem cinco características importantes:

Finitude:
: Um algoritmo deve sempre terminar após um número finito de passos.

Definição:
: Cada passo de um algoritmo deve ser precisamente definido. As ações devem ser definidas rigorosamente e sem ambiguidades.

Entradas:
: Um algoritmo deve ter zero ou mais entradas, isto é informações que são lhe são fornecidas antes do algoritmo iniciar.

Saídas:
: Um algoritmo deve ter uma ou mais saídas, isto é quantidades que tem uma relação específica com as entradas.

Efetividade:
: Um algoritmo deve ser efetivo. Isto significa que todas as operações devem ser suficientemente básicas de modo que possam ser em princípio executadas com precisão em um tempo finito por um humano usando papel e lápis.

É claro que todos nós sabemos construir algoritmos. Se isto não fosse verdade, não conseguiríamos

sair de casa pela manhã, ir ao trabalho, decidir qual o melhor caminho para chegar a um lugar, voltar para casa, etc. Para que tudo isto seja feito é necessário uma série de entradas do tipo: a que hora acordar, que hora sair de casa, qual o melhor meio de transporte, etc.

Um fator importante é pode haver mais de um algoritmo para resolver um problema. Por exemplo, para ir de casa até o trabalho, posso escolher diversos meios de transportes em função do preço, conforto, rapidez, etc. A escolha será feita em função do critério que melhor se adequar as nossas necessidades.

Um exemplo de algoritmo pode ser as instruções que um professor de ginástica passa aos seus alunos em uma academia de ginástica. Por exemplo:

1. Repetir 10 vezes os quatro passos abaixo:
    1. Levantar e abaixar braço direito;
    2. Levantar e abaixar braço esquerdo;
    3. Levantar e abaixar perna esquerda;
    4. Levantar e abaixar perna direita.

Para mostrar outro exemplo de algoritmo considere o seguinte problema. Dispomos de duas vasilhas com capacidades de 9 e 4 litros respectivamente. As vasilhas não tem nenhum tipo de marcação, de modo que não é possível ter medidas como metade ou um terço. Mostre uma seqüência de passos, que usando as vasilhas de 9 e 4 litros encha uma terceira vasilha de medida desconhecida com seis litros de água

Uma possível solução é:

1. Encha a vasilha de 9 litros;
2. Usando a vasilha de 9 litros, encha a vasilha de 4 litros;
3. Despeje o que sobrou na vasilha de 9 litros (5 litros) na terceira vasilha. Observe que falta um litro para completar os seis litros;
4. Esvazie a vasilha de 4 litros;
5. Torne a encher a vasilha de 9 litros;
6. Usando a vasilha de 9 litros encha a vasilha de 4 litros;
7. Esvazie a de 4 litros;
8. Usando o que restou na vasilha de 9 litros (5 litros), encha novamente a vasilha de quatro litros;
9. Despeje o que sobrou na vasilha de 9 litros (1 litro) na terceira vasilha, que agora tem 6 litros.

Um outro exemplo de algoritmo é o que resolve o seguinte problema. Quatro rãs estão posicionadas em cinco casas da seguinte maneira:

	rã 1	rã 2	rã 3	rã 4

As rãs foram treinadas para trocar de casas, mas sempre obedecendo as seguintes regras:

- elas podem pular para a casa vizinha (frente ou trás), se ela estiver vazia;
- elas podem pular sobre a rã vizinha para uma casa livre (frente ou trás).

Mostre como as rãs podem chegar a seguinte posição final:

	rã 4	rã 3	rã 2	rã 1

Este é um problema de colocar em ordem decrescente (ordenação), tarefa muito comum em computação.

Uma possível solução para este problema é a seguinte:

	rã 1	rã 2	rã 3	rã 4
rã 2	rã 1		rã 3	rã 4
rã 2		rã 1	rã 3	rã 4
rã 2	rã 3	rã 1		rã 4
rã 2	rã 3		rã 1	rã 4
rã 2	rã 3	rã 4	rã 1	
rã 2	rã 3	rã 4		rã 1
rã 2	rã 3		rã 4	rã 1
	rã 3	rã 2	rã 4	rã 1
rã 3		rã 2	rã 4	rã 1
rã 3	rã 4	rã 2		rã 1
rã 3	rã 4		rã 2	rã 1
	rã 4	rã 3	rã 2	rã 1

A partir de alguns exemplos anteriores podemos ver que a criação de algoritmos tem muito de inspiração. No entanto, a aparente desordem esconde alguma ordem. Observe que no problema das rãs, elas seguiram um plano básico, que era fazer com que cada uma delas, uma por vez, achasse a sua posição final. A primeira a ir para sua posição final foi a rã 1, em seguida a rã 2 e assim por diante.

Representação de Algoritmos

As formas mais comuns de representação de algoritmos são as seguintes:

Linguagem Natural
: Os algoritmos são expressos diretamente em linguagem natural, como nos exemplos anteriores.

Fluxograma Convencional
: Esta é um representação gráfica que emprega formas geométricas padronizadas para indicar as diversas ações que devem ser executadas e decisões que devem ser tomadas para resolver o problema.

Pseudo-linguagem
: Emprega uma linguagem intermediária entre a linguagem natural e uma linguagem de programação para descrever os algoritmos.

Não existe consenso entre os especialistas sobre qual seria a melhor maneira de representar um algoritmo. Atualmente a maneira mais comum de representar-se algoritmos é através de uma pseudo-linguagem ou pseudo-código. Esta forma de representação tem a vantagem de fazer com que o algoritmo seja escrito de uma forma que está próxima de uma linguagem de programação de computadores. Nesta apostila empregaremos preferencialmente a pseudo-linguagem.

Linguagem Natural

Quase todos os algoritmos que apresentamos até este ponto foram escritos em linguagem natural, ou seja, no nosso caso, habitantes do Brasil, o portugês. O professor de ginástica instruiu seus alunos usando comandos em português. O quebra cabeças das vasilhas de água foi resolvido e explicado por meio de instruções dadas, novamente, em português. Somente no caso das rãs saltadoras preferimos mostrar os seus pulos por meio de uma espécie de tabela.

Fluxogramas

Esta forma de representação de algoritmos emprega várias formas geométricas para descrever cada uma das possíveis açoes durante a execução do algoritmos. Existem algumas formas geométricas que são empregadas normalmente e que estão mostradas na Figura abaixo. Cada uma destas formas se aplica a uma determinada ação como está indicado. Existem outras formas que podem ser aplicadas, no entanto nesta apostila estas formas serão suficientes para os exemplos que serão mostrados.

Como primeiro exemplo de um algoritmo descrito por meio de fluxogramas vamos considerar o exemplo do algoritmo para decidir o que fazer em um dia de domingo. A Figura a seguir mostra o fluxograma equivalente à descrição feita por meio da linguagem natural.

Outro exemplo de um algoritmo descrito por meio de fluxogramas é o problema de calcular a solução da equação de primeiro grau

$$ax+b=0$$

que vale

$$x=-(b/a)$$

se a for diferente de zero. A Figura abaixo mostra um possível algoritmo para resolver este problema.

Pseudo Linguagem

Este modo de representar algoritmos procura empregar uma linguagem que esteja o mais próximo possível de uma linguagem de programação de computadores de alto nível mas evitando de definir regras de construção gramatical muito rígidas. A idéia é usar as vantagens do emprego da linguagem natural, mas restringindo o escopo da linguagem. Normalmente estas linguagens são versões ultra reduzidas de linguagens de alto nível do tipo Pascal ou C. No próximo capítulo veremos um exemplo de uma destas pseudo-linguagens.

Técnicas de Construção de Algoritmos

Neste item iremos apresentar algumas técnicas empregadas na construção de algoritmos. Estas técnicas permitem construir e criar com economia de recursos algoritmos mais eficientes.

Como exemplo iremos usar uma receita que é um algoritmo descrito em lingugem natural. Para economizar texto e facilitar a apresentação a receita não mostra as quantidades dos ingredientes (as entradas). De qualquer maneira estas quantidades não são importantes para nosso estudo de algoritmos. Afinal não estamos em um curso de mestre cucas. Alguns maldosos dizem que o real motivo é que o cozinheiro não quis divulgar o seu segredo. A receita é a seguinte:

- Filé de peixe com molho branco.
    - {preparo dos peixes}
        - Lave os filés e tempere com o suco dos limões, sal, pimenta e salsinha picada. Deixe por 1/2 hora neste tempero. Enxugue e passe cada filé na farinha de trigo. Depois passe pelos ovos batidos e frite na manteiga até ficarem dourados dos dois lados.
    - {preparo do molho branco}
        - Coloque numa panela a manteiga, a farinha e o leite e misture bem. Em fogo médio, cozinhe até engrossar. Adicione o sal, a pimenta e o queijo. Continue com a panela no fogo, cozinhando até que o queijo derreta, mexendo constantemente.
    - {juntando os dois}
        - Adicione queijo parmesão ralado e queijo gruyère. Misture e ponha sobre os filés.
- Fim da receita do filé de peixe com molho branco.

Observe que a receita foi subdividida em partes: preparo dos peixes, preparo do molho branco e finalmente juntar as duas partes. Esta é uma técnica comum na resolução de problemas: dividir para conquistar. A idéia é dividir uma grande tarefa no maior número possível de tarefas simples. Quanto mais simples forem estas tarefas mais facilmente iremos resolvê-los e mais segurança teremos que encontramos uma solução correta.

Vamos considerar agora uma outra receita que tenha molho branco como parte, para ilustrar uma outra técnica comum da criação e descrição de algoritmos. A próxima receita é esta:

- Alface com molho branco.
    - {preparo do molho branco}
        - Coloque numa panela a manteiga, a farinha e o leite e misture bem. Em fogo médio, cozinhe até engrossar. Adicione o sal, a pimenta e o queijo. Continue com a panela no fogo, cozinhando até que o queijo derreta, mexendo constantemente.
    - {preparo da alface}
        - Derreta a manteiga. Junte a alface cortada. Salpique o sal e deixe cozinhar por uns 5 a 10 minutos ou até a alface ficar tenra, ou o líquido da panela secar.
    - {juntando os dois}

        - Junte ao molho branco e ao suco de limão. Coloque numa travessa e enfeite em volta com pão torrado cortado em triângulos.
- Fim da receita do alface com molho branco

Imagine que os pratos abaixo fazem parte de um livro de receitas. Observe atentamente as receitas. Perceba que os dois pratos usam molho branco e que, as duas receitas, ensinam ao leitor como preparar molho branco. Imagine que este livro de receitas tem 20 outros pratos ao molho branco. É fácil perceber que este livro terá numerosas páginas uma vez que, provavelmente, outras receitas básicas (molho de tomate, molho de mostarda, etc.) estarão repetidas em vários pontos do livro. Observe agora uma nova maneira de descrever estas duas receitas:

- Molho branco
    - Coloque numa panela a manteiga, a farinha e o leite e misture bem. Em fogo médio, cozinhe até engrossar. Adicione o sal, a pimenta e o queijo. Continue com a panela no fogo, cozinhando até que o queijo derreta, mexendo constantemente.
- Fim da receita do molho branco

- Filé de peixe com molho branco
    - {preparo dos peixes}
        - Lave os filés e tempere com o suco dos limões, sal, pimenta e salsinha picada. Deixe por 1/2 hora neste tempero. Enxugue e passe cada filé na farinha de trigo. Depois passe pelos ovos batidos e frite na manteiga até ficarem dourados dos dois lados.
    - {preparo do molho branco}
        - Prepare a receita básica de molho branco.
    - {juntando os dois}
        - Adicione queijo parmesão ralado e queijo gruyère. Misture e ponha sobre os filés.
- Fim da receita do Filé de peixe com molho branco.

- Alface com molho branco
    - {preparo do molho branco}
        - Prepare o molho branco segundo a receita básica.
    - {preparo da alface}
        - Derreta a manteiga. Junte a alface cortada. Salpique o sal e deixe cozinhar por uns 5 a 10 minutos ou até a alface ficar tenra, ou o líquido da panela secar.
    - {juntando os dois}
        - Junte ao molho branco e ao suco de limão. Coloque numa travessa e enfeite em volta com pão torrado cortado em triângulos.
- Fim da receita do Alface com molho branco.

Observe a economia de linhas de texto que foi possível devido a separação da receita de molho branco das demais. Se o mesmo procedimento for seguido para as demais receitas básicas, é de se esperar que o livro fique mais "fininho" do que antes.

Você pode argumentar que, no método anterior, era mais rápido seguir uma receita. Agora, ao

preparar o peixe ao molho branco, por exemplo, você tem de interromper a leitura, marcar a página onde você estava, abrir na página da receita de molho branco, aprender a prepará-lo e, então, retornar à receita do peixe. Você tem razão, mas, além da economia de papel, existem outras vantagens em separar a receita do molho branco. Imagine, por exemplo, que amanhã você descubra que o molho branco fica uma delícia se levar uma pitada de alho. Basta modificar a receita de molho branco, que aparece em um único lugar no livro, e todas as receitas "ao molho branco" estarão automaticamente modificadas. No método anterior, seria preciso modificar todas as receitas que usam molho branco, com o risco considerável de esquecermos de modificar alguma delas.

Observe ainda a variação abaixo da receita do peixe:

- Filé de peixe com molho branco
    - {preparo dos peixes}
        - Lave os filés e tempere com o suco dos limões, sal, pimenta e salsinha picada. Deixe por 1/2 hora neste tempero. Enxugue e passe cada filé na farinha de trigo. Depois passe pelos ovos batidos e frite na manteiga até ficarem dourados dos dois lados.
    - {preparo do molho branco}
        - Compre molho branco no supermercado
    - {juntando os dois}
        - Adicione queijo parmesão ralado e queijo gruyère. Misture e ponha sobre os filés.
- Fim da receita do filé de peixe com molho branco.

Você prestou atenção? Ao invés de ensinar a preparar molho branco, a receita instrui você a comprá-lo pronto, no supermercado. Ou, em outras palavras, é possível usar no preparo do seu prato, ingredientes já prontos, preparados por outra pessoa, que você talvez nem conheça. Além disso, se você não é um cozinheiro experiente, o molho à venda no supermercado já foi suficientemente testado e é, provavelmente, gostoso. Embora, nem todos vão concordar com tal infâmia! Você já tem problemas suficientes tentando preparar um bom peixe. Talvez seja melhor usar o molho do supermercado e não ter de se preocupar com essa parte do problema. O uso de algoritmos criados por outros é muito comum na informática e pode reduzir consideravelmente o tempo de criação de um sistema.

Toda a discussão acima tem uma forte analogia com o estudo de algoritmos e técnicas de programação. Isto ficará mais claro para você mais tarde, quando estudarmos procedimentos e funções.

Para ilustrar mais um conceito importante de algoritmos vamos analisar mais um exemplo, considerando o problema de calcular a área de uma mesa retangular. Este cálculo pode ser efetuado se seguirmos os seguintes passos.

- Cálculo da área de uma mesa.
    - Medir a largura da mesa e anotar o resultado.
    - Medir o comprimento da mesa e anotar o resultado.
    - Multiplicar o comprimento pela largura e anotar o resultado.

        - O valor da área da mesa é o resultado anotado no passo anterior.
- Fim do cálculo da área da mesa.

Vamos supor agora que dispomos de uma mesa e de uma toalha cobrindo esta mesa e gostaríamos de saber as áreas da toalha e da mesa. O algoritmo pode ser escrito da seguinte maneira.

- Cálculo das áreas de uma mesa e de uma toalha.
    - Cálculo da área de uma mesa.
        - Medir a largura da mesa e anotar o resultado.
        - Medir o comprimento da mesa e anotar o resultado.
        - Multiplicar o comprimento pela largura e anotar o resultado.
        - O valor da área da mesa é o resultado anotado no passo anterior.
    - Fim do cálculo da área da mesa.
    - Cálculo da área da toalha.
        - Medir a largura da toalha e anotar o resultado.
        - Medir o comprimento da mesa e anotar o resultado.
        - Multiplicar o comprimento pela largura e anotar o resultado.
        - O valor da área da toalha é o resultado anotado passo anterior.
    - Fim do cálculo da área da toalha.
- Fim do cálculo das áreas da mesa e da toalha.

Note que os algoritmos para cálculo da área da mesa e da toalha são parecidos, trazendo a idéia que existe um algoritmo mais geral que é o de cálculo da área de um objeto retangular. Este algoritmo mais geral pode ser então aplicado tanto à mesa como à toalha. Este algoritmo mais geral poderia ser descrito da seguinte maneira:

- Cálculo da área de um objeto retangular ou quadrado.
    - Medir a largura do objeto e anotar o resultado.
    - Medir o comprimento do objeto e anotar o resultado
    - Multiplicar o comprimento pela largura e anotar o resultado.
    - O valor da área e o resultado anotado no passo anterior.
- Fim do cálculo da área de um objeto retangular ou quadrado.

O algoritmo para cálculo da área da mesa e da toalha poderia ser reescrito de forma mais compacta com as seguintes instruções:

- Calcular a área da mesa usando o algoritmo acima.
- Calcular a área da toalha usando o mesmo algoritmo.

Deste modo um mesmo algoritmo pode ser aplicado a diferentes objetos, reduzindo o número de algoritmos a serem definidos.

A maioria dos algoritmos contém decisões, por exemplo, para atravessar uma rua preciso verificar se o sinal de pedestres está verde e verificar se nenhum carro está avançando o sinal, somente após decidir se estes fatos se confirmaram poderei atravessar a rua.

Para considerar um algoritmo que inclua decisões vamos estudar um algoritmo que nos ajude a

decidir o que fazer em um domingo. Um possível algoritmo poderia ser o seguinte:

- Algoritmo de domingo.
    - Acordar.
    - Tomar o café.
    - Se estiver sol vou à praia senão leio o jornal.
    - Almoçar.
    - Ir ao cinema.
    - Fazer uma refeição.
    - Ir dormir.
- Final do domingo.

A possibilidade de tomada de decisões é a característica mais importante de qualquer linguagem de programação. Ela permite que ao computador simular aproximadamente uma característica humana que é a escolha de opções. Sem esta característica o computador seria pouco mais do que uma veloz máquina de calcular.

Vamos agora considerar um exemplo um pouco mais matemático e estudar o algoritmo para calcular as raízes de uma equação do segundo grau da forma

$$ax^2+bx+c=0$$

As raízes podem ser calculadas pelas fórmulas

$$x_1=[-b+(b^2-4ac)^{(1/2)}]/(2a)$$

$$x_2=[-b-(b^2-4ac)^{(1/2)}]/(2a)$$

Aparentemente o algoritmo se reduziria ao cálculo da fórmula, no entanto ao detalharmos as ações devemos prever tudo que pode acontecer durante o cálculo desta fórmula. Por exemplo o que fazer se o valor do coeficiente a for igual a zero? Um possível algoritmo é o seguinte:

- Algoritmo para cálculo de uma equação do segundo grau.
    - Obter os coeficientes a, b e c
    - Se o coeficiente a for igual a zero informar que esta não é uma equação do segundo grau e terminar o algoritmo.
    - Caso contrário continue e faça
        - Calcular delta=b^2-4ac
        - Se o valor de delta for negativo informar que a equação não tem raizes reais e terminar o algoritmo.
        - Caso contrário continue e faça
            - Calcular a raiz quadrada de delta e guardar o resultado como raiz
            - Calcular x_1=(-b + raiz)/(2a)
            - Calcular x_2=(-b - raiz)/(2a)

        - Fornecer como resultado x_1 e x_2
        - Terminar o algoritmo.
- Fim do algoritmo para cálculo de uma equação do segundo grau.

Neste algoritmo em diversos pontos tivemos de tomar decisões e indicar o que fazer em cada uma das possibilidades, mesmo que seja mostrar que não podemos continuar o algoritmo. Toda vez que decisões tiverem de ser tomadas devemos incluir todas as possibilidades para o evento que estamos considerando.

Este é um dos possíveis algoritmos por diversas razões. Por exemplo, poderíamos incluir no algoritmo o cálculo das raízes imaginárias ou no caso do coeficiente *a* ser igual a zero calcular como se fosse uma equação do primeiro grau.

Data da última atualização: 01/09/2001

Tipos de Dados

Objetivos do Capítulo

Os objetivos deste capítulo são apresentar tipos de dados manipulados pelos computadores. Mostraremos que nem todos os dados que manipulamos em nossos cálculos são representados nos computadores. Veremos também que mesmo no caso dos dados que os computadores conseguem manipular, há limitações sérias.

Introdução

Os algoritmos irão manipular dados, que normalmente são fornecidos pelos usuários, e entregar resultados para estes usuários. Uma pergunta importante neste momento é: que tipo de dados poderemos manipular? As linguagens de programação normalmente estabelecem regras precisas para definir que tipos de dados elas irão manipular. A pseudo-linguagem que iremos empregar também estabelece, ainda que informalmente, algumas regras que reduzem o conjunto de dados existentes na natureza a um conjunto mais simples e somente este conjunto poderá ser manipulado pelos algoritmos.

O objetivo deste capítulo é mostrar que tipos de dados poderão ser manipulados nos algoritmos que iremos criar. Verificaremos que computadores tem limitações quanto aos tipos de dados numéricos que podem manipular, mas que também podem manipular outros tipos de dados não numéricos com facilidade.

Existem três tipos básicos de dados que iremos manipular nos algoritmos que iremos criar:

- Dados numéricos
- Dados literais ou alfa-numéricos
- Dados lógicos

Cada um destes tipos de dados será detalhado nas próximas seções.

Dados Numéricos

O conjunto de dados mais comuns é o de números naturais, que é representado por **N**. Este conjunto é definido como

$$N = \{0,1,2,3,...\}$$

Este conjunto de números é usado quando queremos falar sobre o número de amigos que temos ou quantos CDs musicais temos na nossa coleção. Embora seja fácil imaginar que um pastor de ovelhas há 3000 anos atrás pudesse usar este conjunto com facilidade, é bom lembrar que o conceito do número zero é difícil de ser entendido. O conjunto dos números naturais usado pelos primeiros seres humanos não incluía o zero. Os pastores sabiam que se 20 ovelhas tinham ido para os campos, eles tinha de esperar pelas mesma 20 ovelhas na volta à noite. Agora se não havia ovelhas para que precisaríamos contar, somar ou subtrair? A natureza tem horror ao vácuo e o nada é um conceito complicado para os seres humanos. O número zero é uma invenção dos matemáticos hindus e é recente, sendo introduzido em XXXX aproximadamente.

O conjunto dos números naturais é subconjunto dos conjunto dos números inteiros que é definido como

$$Z = \{...,-3,-2,-1,0,+1,+2,+3,...\}$$

Aqui estamos falando de conceitos mais complicados que hoje em dia já fazem parte do nosso dia a dia. Estamos no domínio dos números relativos, que incluem os números positivos, o zero e os números negativos. A mente humana teve de atingir graus de abstração maiores para imaginar

operações que incluiam números negativos. Hoje em dia não é necessário ser um matemático pós-graduado para trabalhar facilmente com este tipo de números. Todos nós somos capazes de perceber que uma temperatura de -3 graus centígrados e mais fria do que +3 graus. Quem não fica preocupado quando sabe que sua conta corrente no banco está com saldo de -300 reais. Rapidamente iremos tentar transformar este número negativo em número positivo, e o que mais importante sabemos que precisamos de pelo menos 300 reais para sair do vermelho.

O próximo conjunto na nossa curta viagem pelos domínios dos números é o conjunto dos números fracionários, que é representado por **Q**. Este conjunto é composto por todos os números que podem ser escritos como uma fração da forma *p/q* onde *p* e *q* pertencem ao conjunto dos números inteiros. Este conjunto pode ser definido como:

$$Q = \{p/q \mid p, q \text{ pertencem a } Z\}$$

Continuando nesta excursão vamos para o conjunto dos números reais (R) que é a união do conjunto dos números fracionários e o dos números irracionais. Números irracionais não são os números que não conseguem pensar, mas sim aqueles que não podem ser expressos por uma fração *p*/q. Um exemplo muito conhecido de número irracional é o PI que é igual a 3.14159...

O último conjunto é o dos números complexos. Neste conjunto os números são representados da seguinte maneira

$$n = a + ib$$

Os números *a* e *b* são números reais e i representa a raiz quadrada do número inteiro -1. Quando *b* é igual a 0 o número é um número pertencente ao conjunto dos reais. Como pode-se ver, a medida que fomos passando de um conjunto para outro, aumentou o nível de abstração das quantidades que os números pertencentes aos conjuntos representam.

A Figura a seguir é uma representação das relações de pertinência entre os conjuntos de números que analisamos até aqui.

Os dados numéricos que os algoritmos que iremos criar e que a maioria dos computadores manipulam são de dois tipos:

- Dados inteiros
- Dados reais

Neste ponto é importante assinalar dois fatos importantes. Primeiro computadores trabalham com uma base diferente da base que usamos todos dias que é a base 10. Computadores usam a base 2, e no processo de conversão entre bases podem ocorrer problemas de perda de dígitos significativos. Por exemplo, o número real 0.6 ao ser convertido para a base dois gera uma dízima periódica.

Outro fato importante é que a largura das palavras de memória do computador é limitada e portanto o número de dígitos binários que podem ser armazenados é função deste tamanho. Isto é similar ao que aconteceria se tivéssemos que limitar o número o dígitos decimais que usamos para representar os números de nossas contas no banco. Por exemplo, assuma que só podemos armazenar quantias com 6 dígitos inteiros e dois decimais, deste modo se ganhássemos na loteria R$ 1.000.000,00 não

seria possível representar este valor no extrato do banco. Portanto no processo de conversão e desconversão entre bases pode haver perda de informação.

Dados Numéricos Inteiros

O conjunto dos dados inteiros pode ser definido como

$$Z=\{...,-3,-2,0,1,2,...\}.$$

As linguagens usadas para programar computadores são muito exigentes com a maneira com que os dados são representados. Por esta razão vamos passar a definir como deveremos representar os dados nos algoritmos. Os dados inteiros tem a seguinte forma:

NúmeroInteiro = [+,-]algarismo{algarismo}

Neste texto iremos usar uma notação especial para definir como iremos representar os elementos da linguagem. A medida que formos apresentando os elementos a notação será também explicada. No caso da definição dos dados inteiros temos os seguintes elementos. O elemento básico é o algarismo que é um dos seguintes caracteres:

algarismo = 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Os elementos entre colchetes são opcionais. Portanto, o sinal de + e - entre colchetes significa que um número inteiro pode ou não ter sínal. Em seguida temos um algarismo que é obrigatório. Isto é dados inteiros tem de ter pelo menos um algarismo. A seguir temos a palavra algarismo entre chaves, o que significa que um número inteiro deve ter pelo menos um algarismo e pode ser seguido por uma sequência de algarismos. Deste modo elementos que aparecem entre chaves são elementos que podem ser repetidos.

São portanto exemplos de números inteiros:

a) `+3`
b) `3`
c) `-324`

Como exemplos de formas erradas de representação de números inteiros, temos:

a) `+3.0` Não é possível usar ponto decimal
b) `+ 123` Espaços em branco não são permitidos

Dados Numéricos Reais

Os dados reais tem a seguinte forma:

[+,-]algarismo{algarismo}.algarismo{algarismo}.

Ou seja um número real pode ou não ter sinal, em seguida um conjunto de pelo menos um algarismo, um ponto decimal e depois um conjunto de pelo menos um algarismo. É importante notar que o separador entre a parte inteira e a fracionário é o ponto e não a vírgula.

São exemplos de números reais:

a) `0.5`
a) `+0.5`
a) `-3.1415`

Abaixo mostramos exemplos de formas erradas de representação de números reais, temos:

a) `+3,0` Vírgula não pode ser separador entre as partes real e inteira
b) `0.333...` Não é possível usar representação de dízimas

Dados Literais

Dados literais servem para tratamento de textos. Por exemplo, um algoritmo pode necessitar de imprimir um aviso para os usuários, ou um comentário junto com um resultado numérico. Outra possibilidade é a necessidade de ler dados tais como nomes, letras, etc.

Este tipo de dados pode ser composto por um único caracter ou por um conjunto de pelo menos um destes elementos. Conjuntos são conhecidos como cadeias de caracteres, tradução da expressão em inglês, "*character string*".

Um ponto importante que deve ser abordado agora é o que se entende por caractere. Caracteres são basicamente as letras minúsculas, maiúsculas, algarismos, sinais de pontuação, etc. Em computação caracteres são representados por códigos binários e o mais disseminado de todos é o código ASCII. Este padrão foi definido nos Estados Unidos e é empregado pela quase totalidade dos fabricantes de computadores e programas. O apêndice mostra a tabela ASCII com estes códigos.

Os caracteres que normalmente são empregados nos algoritmos são os seguintes:

Letras maiúsculas:
```
A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P
| Q | R | S | T | U | V | W | X | Y | Z
```
Letras minúsculas:
```
a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p
| q | r | s | t | u | v | w | x | y | z
```
Algarismos:
```
0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```
Caracteres de pontuação:
```
; | : | ! | ? | * | ( | ) | \ | / | + | - | = | < | >
```

Constantes Caracter

Caracteres podem aparecer sozinhos, e neste caso são chamados de constante caracter e são representados entre o caracter '. Abaixo mostramos exemplos de constantes caracter:

- 'a'
- 'A'
- ';'
- '+'

Cadeias de Caracter

Cadeias de caracteres são conjuntos de um ou mais caracteres e são cercados pelo caracter ". Por exemplo:

- "Linguagem de programação"
- "Qual é o seu nome?"
- "12345"

Dados Lógicos

Este tipo de dados é intensamente aplicado durante o processo de tomada de decisões que o computador frequentemente é obrigado a fazer. Em muitos textos este tipo de dados também é chamado de dados booleanos, devido a George Boole, matemático que deu ao nome à álgebra (álgebra booleana) que manipula este tipo de dados. Os dados deste tipo somente podem assumir dois valores: **verdadeiro** e **falso**.

Computadores tomam decisões, durante o processamento de um algoritmo, baseados nestes dois valores. Por exemplo, considere a sentença a seguir que é um caso típico de decisão que o computador é capaz de tomar sem intervenção humana.

Se está chovendo então procurar guarda-chuva.

Nesta sentença temos a questão lógica "Se está chovendo". Esta expressão somente pode ter como resultado um de dois valores: verdade ou falso. Nos nossos algoritmos estes valores serão representados por `verdade` e `falso`. Mais adiante ficará claro como este tipo de dados será empregado nos algoritmos.

Exercícios

1. Indique quais são os números inteiros válidos e os inválidos. Justifique suas respostas.
    1. `1234`
    2. `1234.0`
    3. `1234,0`
    4. `+12`
    5. `- 356`
    6. `234`
2. Indique quais são os números erais válidos e os inválidos. Justifique suas respostas.
    1. `1234.5`
    2. `+ 1234.0`
    3. `1234,0`
    4. `+12`
    5. `-.356`

6. `234/3`

3. Represente o seu nome como uma cadeia de caracteres.
4. Represente a letra s como uma constante caracter.

Data da última atualização: 20/09/2001

Variáveis

Objetivos do Capítulo

Este capítulo

Introdução

Este capítulo

Variáveis

São os nomes que utilizamos para referenciar as posições de memória. Como já foi mostrado no capítulo de Introdução, a memória de um computador pode ser entendida como um conjunto ordenado e numerado de palavras. Na maioria dos PCs que usamos diariamente a memória pode ser considerada como um conjunto ordenado e numerado de bytes (8 bits). As linguagens de programação de alto nível atribuem nomes as posições de memória que armazenam os dados a serem processados. Deste modo os programadores têm mais facilidade para construir seus algoritmos.

Na linguagem a-- um nome de variável é contruído da seguinte maneira: uma letra seguida por um conjunto de letras ou algarismos. Por exemplo, os nomes seguintes são nomes de variáveis válidos:

- i

- valor
- nome
- nota1

Como nomes inválidos podemos dar os seguintes exemplos:

- 2nota (nome começado por algarismo)
- nome de aluno (nome com espaços em branco no meio)

Durante a apresentação da linguagem iremos fazer referências a listas de variáveis. Uma lista de variáveis é um conjunto de nomes de variáveis separados por vírgulas, por exemplo: Ex. nota1, nota2, media

Na linguagem a-- uma variável não precisa ser definida antes de ser usada em um algoritmo. A variável irá assumir, dinamicamente, o tipo do dado que estiver sendo atribuído a esta variável. Por exemplo se armazenarmos o valor 5 em uma variável ela passará a ser do tipo inteiro. Caso resolvamos trocar o valor para 3.14 a variável passará a ser real.

Data da última atualização: 16/08/2001

Variáveis

Introdução

Variáveis

São os nomes que utilizamos para referenciar as posições de memória. Como já foi mostrado no capítulo de Introdução, a memória de um computador pode ser entendida como um conjunto ordenado e numerado de palavras. Na maioria dos PCs que usamos diariamente a memória pode ser

considerada como um conjunto ordenado e numerado de bytes (8 bits). As linguagens de programação de alto nível atribuem nomes as posições de memória que armazenam os dados a serem processados. Deste modo os programadores tem mais facilidade para construir seus algoritmos.

Na linguagem a-- um nome de variável é contruído da seguinte maneira: uma letra seguida por um conjunto de letras ou algarismos. Por exemplo, os nomes seguintes são nomes de variáveis válidos:

- i
- valor
- nome
- nota1

Como nomes inválidos podemos dar os seguintes exemplos:

- 2nota (nome começado por algarismo)
- nome de aluno (nome com espaços em branco no meio)

Durante a apresentação da linguagem iremos fazer referências a listas de variáveis. Uma lista de variáveis é um conjunto de nomes de variáveis separados por vírgulas, por exemplo: Ex. nota1, nota2, media

Na linguagem a-- uma variável não precisa ser definida antes de ser usada em um algoritmo. A variável irá assumir, dinamicamente, o tipo do dado que estiver sendo atribuído a esta variável. Por exemplo se armazenarmos o valor 5 em uma variável ela passará a ser do tipo inteiro. Caso resolvamos trocar o valor para 3.14 a variável passará a ser real.

Expressões

Uma vez que já temos os dados e as variáveis podemos passar ao próximo estágio que seria a criação de expressões. No entanto, para que a expressão possa estar completa precisamos de operadores que possam ser aplicados a estes dados. Os operadores da linguagem a-- são basicamente os mesmos encontrados em outras linguagens de programação.

Na linguagem a-- existem basicamente três tipos de expressões:

- Expressões Aritméticas;
- Expressões Lógicas;
- Expressões Literais.

Cada um deste tipos tem os seus operadores próprios.

Expressões Aritméticas

Expressões aritméticas são aquelas que apresentam como resultado um valor numérico que pode ser um número inteiro ou real, dependendo dos operandos e operadores. Os operadores aritméticos disponíveis em a-- estão mostrados na Tabela a seguir.

Operador	Descrição	Prioridade
+	**Soma**	**3**
-	**Subtração**	**3**
*	**Multiplicação**	**2**
/	**Divisão**	**2**
mod	**Módulo (Resto da divisão inteira)**	**2**
+	**Operador unário (sinal de mais)**	**1**
-	**Operador unário (sinal de menos)**	**1**

A prioridade indica a ordem em que cada operação deverá ser executada. Quanto menor o número maior a prioridade da operação. Observe que o operador de multiplicação é o caracter asterisco, um símbolo que é empregado na maioria das linguagens para esta operação.

Expressões aritméticas podem manipular operandos de dois tipos: reais e inteiros. Se todos os operandos de uma expressão são do tipo inteiro então a expressão fornece como resultado um número inteiro. Caso pelo menos um dos operandos seja real o resultado será real. Isto pode parecer estranho a princípio, mas este procedimente reflete a forma como as operações são executadas pelos processadores. Por exemplo o resultado da operação 1/5 é 0, porque os dois operadores são inteiros. Caso a expressão tivesse sido escrita como 1.0/5 então o resultado 0.2 seria o correto. A seguir mostramos exemplos de algumas expressões aritméticas:

- A+B-C
- a/b
- 3.14*(A+B)

Observar que as expressões somente podem ser escritas de forma linear, isto é o sinal de divisão é uma barra inclinada. Portanto frações somente podem ser escritas conforme o exemplo acima (a/b). Outro ponto importante é a ordem de avaliação das expressões, as prioridades mostradas na Tabela dos operadores não é suficiente para resolver todas as situações e precisamos apresentar algumas regras adicionais:

1. Deve-se primeiro observar a prioridade dos operadores conforme a Tabela dos operadores, ou seja operadores com maior prioridade (números menores) são avaliados primeiro. Caso haja empate na ordem de prioridade resolver a expressão da esquerda para a direita.
2. Parênteses servem para mudar a ordem de prioridade de execução das operações. Quando houver parênteses aninhados (parênteses dentro de parênteses) as expressões dentro dos mais internos são avaliadas primeiro.

Vamos considerar alguns exemplos para mostrar como estas regras são aplicadas. Considere as seguintes variáveis:

- A=2.0
- B=4.0
- C=1.0

Vamos então analisar expressões com estas variáveis e seus resultados.

1. A*B-C
2. A*(B-C)
3. B+A/C+5
4. (B+A)/(C+5)

A primeira expressão tem como resultado o valor 7, como era de se esperar. Na segunda expressão a ordem de avaliação é alterada pelo parênteses e primeiro é feita a subtração e o resultado passa ser 6. A primeira operação na terceira expressão é a divisão que tem maior prioridade. Neste caso o resultado final é 11. Na última expressão as somas são realizadas primeiro e por último a divisão, ficando o resultado igual a 1.

Expressões Lógicas

Expressões lógicas são aquelas cujo resultado pode somente assumir os valores verdadeiro ou falso. Os operadores lógicos e sua ordem de precedência são mostrados na Tabela a seguir.

Operador	**Descrição**	**Prioridade**
ou	**Ou lógico**	**3**
e	**E lógico**	**2**
não	**Não lógico**	**1**

Estes operadores e seus dados também possuem uma espécie de tabuada que mostra os resultados de operações básicas. A Tabela a seguir mostra os resultados da aplicação destes operadores à duas variáveis lógicas.

A	**B**	**A ou B**	**A e B**	**não A**
falso	falso	falso	falso	verdade
falso	verdade	verdade	falso	verdade
verdade	falso	verdade	falso	falso
verdade	verdade	verdade	verdade	falso

Há ainda um outro tipo de operadores que podem aparecer em expressões lógicas que são os operadores relacionais. Estes operadores estão mostrados na Tabela a seguir.

Operador	**Descrição**
>	**maior que**
<	**menor que**

>=	**maior ou igual a**
<=	menor ou igual a
==	igual a
!=	diferente de

Alguns exmplos de expressões lógicas são:

- (A e B) ou C
- (A < 5) ou (b < 3)

Comandos

Agora iremos apresentar uma série de definições informais dos comandos da linguagem a--.

- **Lista de comandos:**

  <lista de comandos>
  Uma sequência de comandos válidos da linguagem. Os comandos deverão estar separados por ;

- **Comando de atribuição:**
  =
  A expressão do lado direito do operador é calculada e o seu resultado é atribuído à variável do lado esquerdo do sinal.
  Exemplos:
  a = -5.0;
  b = - a + x - 3.0;

- **Leitura dos valores de entrada do algoritmo:**

  ler <lista de variáveis>;
  Valores são lidos do teclado, um de cada vez e atribuídos as variáveis listadas na ordem em que aparecem na lista.
  Exemplos:
  ler nota1, nota2;
  ler a, b, c;

- **Impressão dos resultados dos algoritmos:**

  imprimir <lista de variáveis>
  Imprime os valores das saídas do algoritmo. Os valores das variáveis são impressos,

um de cada vez, na ordem em que aparecem na lista de variáveis.

Exemplo:

imprimir media, nota1, nota2;
imprimir x1, x2;

É possível misturar mensagens para os usuários no comando de impressão. Por exemplo:

```
imprimir "A primeira nota vale", nota1, " e a segunda ", nota2;
```

- **Comando de repetição enquanto:**

enquanto <expressão lógica> **faça**
<lista de comandos>
fim enquanto
Os comandos entre *enquanto* e *fim enquanto* são executados repetidamente enquanto a condição de teste for satisfeita;
Exemplo:

```
i=0;
enquanto i<10 faça
    i=i+1;
fim enquanto;
imprimir i;
```

- **Comando de repetição para:**

para contador = expressão inteira inicial **até** expressão inteira final **faça**
<lista de comandos>
fim para
Os comandos entre *faça* e *fim para* são executados repetidamente. O número de vezes é controlado pela variável contador. Ela começa no valor definido pela expressão inteira inicial e termina no valor definido pela expressão inteira final. Caso o valor inicial seja menor que o valor final, a variável é incrementada de 1, e de -1 no caso contrário.
Exemplo:

```
para i=0 até 10 faça
    imprimir 2*i;
fim para;
```

- **Comando de teste (desvio)**

Se <expressão lógica> então
<lista de comandos1>
senão
<lista de comandos2>
fimse
Testa se a expressão lógica é verdade. Caso seja verdade então executa a lista de comandos 1, senão executa a lista de comandos 2.

Exemplo:

se (i mod 2) == 0 então
imprimir "O número é par.";
senão
imprimr "O número é ímpar."
fimse;

- **Função**

- Base de toda a linguagem a--. Um programa é composto de funções. A primeira função a ser executada é obrigatoriamente chamada de *principal*. O que a função executa está definido pela lista de comandos. A lista de parâmetros, é uma lista de variáveis contendo os valores que passamos para a função usar durante a sua execução. Uma função poderá retornar um valor

***nome da função* (<lista de parâmetros>) início**
<lista de comandos>
fim de *nome da função;*

- **Retorna de valores ao fim da função**

retorna <valor>;
- Caso a função tenha que retornar algum valor para o algoritmo que chamou a função, deve-se usar este comando. Este valor pode ser fornecido através de uma variável ou de uma expressão.
Exemplo:
retorna x;
retorna y + 1;

- **Comentários**

//
- O restante da linha é um comentário explicando o trecho de programa.
/* */
Tudo que estiver entre /* e */ é considerado um comentário. Este tipo de comentário pode se extender por diversas linhas.

Exemplos de Algoritmos

1. **Algoritmo de Euclides**
   Dados dois números positivos m e n encontre seu maior divisor comum, isto é o maior inteiro positivo que divide tanto m como n. Assuma que m é sempre maior que n, e n diferente de zero.
   - **principal () início**

- **ler m, n;**
- **r = m mod n; // resto da divisão de m por n**
- **enquanto r != 0 faça**
    - **m = n;**
    - **n = r;**
    - **r = m mod n;**
- **fim do enquanto**
- **imprimir n;**

- **fim de principal**

Este algoritmo escrito em C pode ser visto no arquivo: au1ex1.c

---

2. **Multiplicação de dois números inteiros positivos**

- **principal () início// achar quanto vale m*n**
    - **ler m, n;**
    - **r = 0;**
    - **enquanto n != 0 faça**
        - **r = r + m;**
        - **n = n-1;**
    - **fim do enquanto**
    - **imprimr r;**
- **fim de principal**

Este algoritmo escrito em C pode ser visto no arquivo: au1ex2.c

---

3. **Resolução de uma equação do segundo grau.**
   Neste algoritmo vamos assumir que o coeficiente a da equação é sempre diferente de 0.

- **principal () início**
    - **ler a, b, c**
    - **delta = b*b-4*a*c**
    - **se delta < 0**
        - **então**
            - **imprimir ¨Não há raizes reais.¨**
        - **senão início**
            - **x1 = (-b + sqrt(delta))/(2*a)**
            - **x2 = (-b + sqrt(delta))/(2*a)**
            - **imprimir x1, x2**
    - **fim de se**
- **fim**

Este algoritmo escrito em C pode ser visto no arquivo: au1ex3.c

Exercícios

Para resolver estes exercícios encorajamos fortemente os alunos a procurarem na biblioteca as referências bibliográficas dadas e outros livros sobre o assunto.

1. **Defina os termos Unidade de Controle e Unidade Aritmética.**
2. **Discuta as diferenças entre um microprocessador e o microcomputador.**
3. **Escreva um programa que apresente a mensagem "Alo mundo" ao usuário.**
   Solução: Alo Mundo
4. **Qual o número exato de bytes em 64Kbytes?**
5. **Defina brevemente Sistema Operacional.**
6. **Qual a primeira função a ser executada em um programa C?**

Apêndice 1 - TABELA ASCII

A Tabela ASCII (American Standard Code for Information Interchange) é usada pela maior parte da industria de computadores para a troca de informações. Cada caracter é representado por um código de 8 bits (um byte). Abaixo mostramos a tabela ASCII de 7 bits. Existe uma tabela extendida para 8 bits que inclue os caracteres acentuados.

Caracter	Decimal	Hexadecimal	Binário	Comentário
NUL	00	00	0000 0000	Caracter Nulo
SOH	01	01	0000 0001	Começo de cabeçalho de transmissão
STX	02	02	0000 0010	Começo de texto
ETX	03	03	0000 0011	Fim de texto
EOT	04	04	0000 0100	Fim de transmissão
ENQ	05	05	0000 0101	Interroga
ACK	06	06	0000 0110	Confirmação
BEL	07	07	0000 0111	Sinal sonoro
BS	08	08	0000 0100	Volta um caracter
HT	09	09	0000 1001	Tabulação Horizontal
LF	10	0A	0000 1010	Próxima linha
VT	11	0B	0000 1011	Tabulação Vertical
FF	12	0C	0000 1100	Próxima Página

CR	**13**	**0D**	**0000 1101**	**Início da Linha**
SO	**14**	**0E**	**0000 1110**	**Shift-out**
SI	**15**	**0F**	**0000 1111**	**Shift-in**
DLE	**16**	**10**	**0001 0000**	**Data link escape**
D1	**17**	**11**	**0001 0001**	**Controle de dispositivo**
D2	**18**	**12**	**0001 0010**	**Controle de dispositivo**
D3	**19**	**13**	**0001 0011**	**Controle de dispositivo**
D4	**20**	**14**	**0001 0100**	**Controle de dispositivo**
NAK	**21**	**15**	**0001 0101**	**Negativa de Confirmação**
SYN	**22**	**16**	**0001 0110**	**Synchronous idle**
ETB	**23**	**17**	**0001 0111**	**Fim de transmissão de bloco**
CAN	**24**	**18**	**0001 1000**	**Cancela**
EM	**25**	**19**	**0001 1001**	**Fim de meio de transmissão**
SUB	**26**	**1A**	**0001 1010**	**Substitui**
ESC	**27**	**1B**	**0001 1011**	**Escape**
FS	**28**	**1C**	**0001 1100**	**Separador de Arquivo**
GS	**29**	**1D**	**0001 1101**	**Separador de Grupo**
RS	**30**	**1E**	**0001 1110**	**Separador de registro**
US	**31**	**1F**	**0001 1111**	**Separador de Unidade**
Espaço	**32**	**20**	**0010 0000**	
!	**33**	**21**	**0010 0001**	
"	**34**	**22**	**0010 0010**	
#	**35**	**23**	**0010 0011**	
$	**36**	**24**	**0010 0100**	
%	**37**	**25**	**0010 0101**	
&	**38**	**26**	**0010 0110**	
'	**39**	**27**	**0010 0111**	
(	**40**	**28**	**0010 1000**	
)	**41**	**29**	**0010 1001**	
*****	**42**	**2A**	**0010 1010**	
+	**43**	**2B**	**0010 1011**	
,	**44**	**2C**	**0010 1100**	
-	**45**	**2D**	**0010 1101**	
.	**46**	**2E**	**0010 1110**	
/	**47**	**2F**	**0010 FFFF**	
0	**48**	**30**	**0011 0000**	
1	**49**	**31**	**0011 0001**	
2	**50**	**32**	**0011 0010**	
3	**51**	**33**	**0011 0011**	
4	**52**	**34**	**0011 0100**	
5	**53**	**35**	**0011 0101**	
6	**54**	**36**	**0011 0110**	

7	**55**	**37**	**0011 0111**	
8	**56**	**38**	**0011 1000**	
9	**57**	**39**	**0011 1001**	
:	**58**	**3A**	**0011 1010**	
;	**59**	**3B**	**0011 1011**	
<	**60**	**3C**	**0011 1100**	
=	**61**	**3D**	**0011 1101**	
>	**62**	**3E**	**0011 1110**	
?	**63**	**3F**	**0011 1111**	
@	**64**	**40**	**0100 0000**	
A	**65**	**41**	**0100 0001**	
B	**66**	**42**	**0100 0010**	
C	**67**	**43**	**0100 0011**	
D	**68**	**44**	**0100 0100**	
E	**69**	**45**	**0100 0101**	
F	**70**	**46**	**0100 0110**	
G	**71**	**47**	**0100 0111**	
H	**72**	**48**	**0100 1000**	
I	**73**	**49**	**0100 1001**	
J	**74**	**4A**	**0100 1010**	
K	**75**	**4B**	**0100 1011**	
L	**76**	**4C**	**0100 1100**	
M	**77**	**4D**	**0100 1101**	
N	**78**	**4E**	**0100 1110**	
O	**79**	**4F**	**0100 1111**	
P	**80**	**50**	**0101 0000**	
Q	**81**	**51**	**0101 0001**	
R	**82**	**52**	**0101 0010**	
S	**83**	**53**	**0101 0011**	
T	**84**	**54**	**0101 0100**	
U	**85**	**55**	**0101 0101**	
V	**86**	**56**	**0101 0110**	
W	**87**	**57**	**0101 0111**	
X	**88**	**58**	**0101 1000**	
Y	**89**	**59**	**0101 1001**	
Z	**90**	**5A**	**0101 1010**	
[	**91**	**5B**	**0101 1011**	
****	**92**	**5C**	**0101 1100**	
]	**93**	**5D**	**0101 1101**	
^	**94**	**5E**	**0101 1110**	
_	**95**	**5F**	**0101 1111**	
`	**96**	**60**	**0110 0000**	

a	**97**	**61**	**0110 0001**	
b	**98**	**62**	**0110 0010**	
c	**99**	**63**	**0110 0011**	
d	**100**	**64**	**0110 0100**	
e	**101**	**65**	**0110 0101**	
f	**102**	**66**	**0110 0110**	
g	**103**	**67**	**0110 0111**	
h	**104**	**68**	**0110 1000**	
i	**105**	**69**	**0110 1001**	
j	**106**	**6A**	**0110 1010**	
k	**107**	**6B**	**0110 1011**	
l	**108**	**6C**	**0110 1100**	
m	**109**	**6D**	**0110 1101**	
n	**110**	**6E**	**0110 1110**	
o	**111**	**6F**	**0110 1111**	
p	**112**	**70**	**0111 0000**	
q	**113**	**71**	**0111 0001**	
r	**114**	**72**	**0111 0010**	
s	**115**	**73**	**0111 0011**	
t	**116**	**74**	**0111 0100**	
u	**117**	**75**	**0111 0101**	
v	**118**	**76**	**0111 0110**	
w	**119**	**77**	**0111 0111**	
x	**120**	**78**	**0111 1000**	
y	**121**	**79**	**0111 1001**	
z	**122**	**7A**	**0111 1010**	
{	**123**	**7B**	**0111 1011**	
|	**124**	**7C**	**0111 1100**	
}	**125**	**7D**	**0111 1101**	
~	**126**	**7E**	**0111 1110**	
DELETE	**127**	**7F**	**0111 1111**	

Data da última atualização: 30/01/97

Fonte:http://equipe.nce.ufrj.br/adriano/algoritmos/apostila/indice.htm