Universidade Federal de Itajubá – UNIFEI
Curso de Engenharia de Computação
ECO 002 – Técnicas de Programação – Prof. Edmilson Marmo Moreira

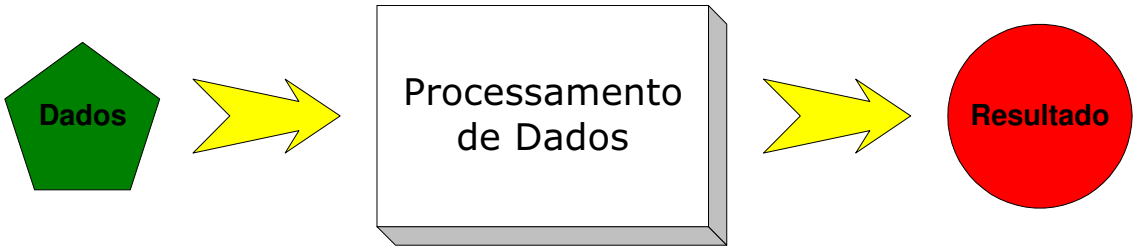
Capítulo 1 – Introdução

"Não basta ensinar ao homem uma especialidade, porque se tornará assim uma máquina utilizável, mas não uma personalidade. É necessário que adquira um sentimento, um senso prático daquilo que vale a pena ser empreendido, daquilo que é belo, do que é moralmente correto. A não ser assim, ele se assemelhará, com seus conhecimentos profissionais, mais a um cão ensinado do que a uma criatura harmoniosamente desenvolvida. Deve aprender a compreender as motivações dos homens, suas quimeras e suas angústias, para determinar com exatidão seu lugar preciso em relação a seus próximos e à comunidade."

Albert Einstein

1.1 Considerações Iniciais

A tarefa de processamento de dados consiste em tomar certa informação, processá-la e obter o resultado desejado.



Desde que o homem começou a processar dados, ele tentou construir máquinas para ajudá-lo em seu trabalho. O computador eletrônico é o resultado dessas tentativas que vêm sendo realizadas através dos séculos.

1.2 Histórico

Década de 1940

1943 Alan Turing estava envolvido no desenvolvimento de uma máquina para decifrar o código das mensagens nazistas, conhecido como *Enigma.*

1944 Foi construído o primeiro computador eletromecânico,

batizado de Mark I. Utilizado pela marinha americana. O Mark I usava os mesmos princípios da Máquina Analítica, criada por Babbage cem anos antes.

1946 O ENIAC (*Eletronic Numerical Integrator and Calculator*) é considerado o primeiro computador digital e eletrônico. Desenvolvido pelos engenheiros J. Presper Eckert e John W. Mauchly, na Universidade da Pensilvânia.

Foi criado para cálculo balístico e, posteriormente, utilizado no projeto da bomba de hidrogênio. Ficou em operação no período de 1946 a 1955.

Possuía 17 mil válvulas, 10 mil capacitores, 70 mil resistores e pesava 30 toneladas. Quando em operação, consumia cerca de 140 quilowatts e era capaz de realizar 5 mil adições por segundo.

Para trabalhar com o ENIAC era necessário conhecer profundamente o funcionamento do *hardware,* pois a programação era feita em painéis, através de 6 mil conectores, utilizando linguagem de máquina. Esta tarefa facilmente poderia levar alguns dias.

Corretamente programado, um cálculo que levasse vinte e quatro horas manualmente era resolvido em menos de trinta segundos.

O professor John von Neumann, consultor do projeto do ENIAC, imaginou uma máquina de propósito geral onde tanto instruções quanto dados fossem armazenados em uma mesma memória, tornando o processo de programação muito mais rápido e flexível (“Arquitetura de von Neumann”).

1949 EDSAC (*Eletronic Delay Storage Automatic Calculator*) foi o primeiro computador a implementar o conceito de “programa armazenado”.

Outros computadores foram construídos nessa mesma época com base no mesmo princípio.

Nesta fase, os computadores ainda não possuíam dispositivos com função de interface com os usuários e o conceito de

sistema operacional.

Década de 1950

O uso do transistor e da memória magnética contribuiu para o enorme avanço dos computadores da época. (2ª geração)

O transistor permitiu o aumento da velocidade e da confiabilidade no processamento, e as memórias magnéticas permitiram o acesso mais rápido aos dados, maior capacidade de armazenamento e computadores menores.

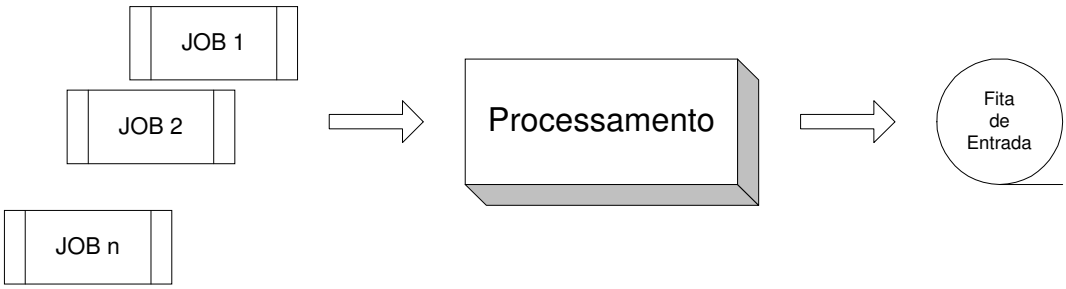
Apesar do invento do transistor datar do final da década de 1940 (*Bell Telephone Laboratories*, 1948), os primeiros computadores transistorizados foram lançados comercialmente apenas no final da década de 1950.

Surgiram as primeiras empresas da indústria de computadores: Raytheon, RCA, Burroughs e a IBM, o que levou à criação dos primeiros computadores para utilização em aplicações comerciais.

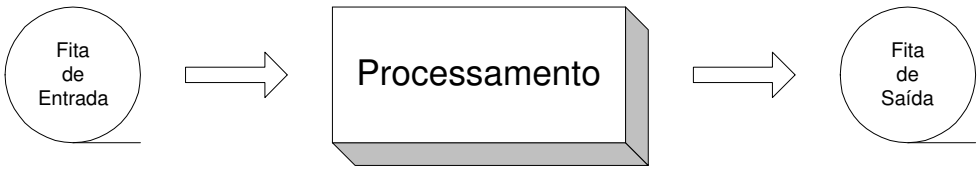
1951 UNIVAC (*Universal Automatic Computer*) foi o primeiro computador bem-sucedido fabricado para fins comerciais (censo dos Estados Unidos).

O Massachusetts Institute of Technology (MIT) colocou em operação o que é considerado o primeiro computador voltado para o processamento em tempo real, o Whirlwind I. Ele introduziu a tecnologia de memória magnética.

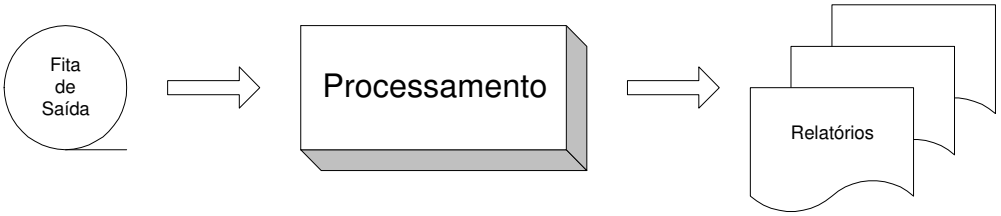
Os programas ou *jobs* passaram a ser perfurados em cartões, que, submetidos a uma leitora, eram gravados em uma fita de entrada.



A fita, então, era lida pelo computador, que executava um programa de cada vez, gravando o resultado do processamento em uma fita de saída.



Ao término de todos os programas, a fita de saída era lida e impressa.



A esse tipo de processamento, onde um conjunto de programas era submetido ao computador, deu-se o nome de *processamento batch*.

1953 Primeiro sistema operacional, chamado monitor, desenvolvido para o computador IBM 701

Surgimento das primeiras linguagens de programação de alto nível, como FORTRAN, ALGOL e COBOL, os programas deixaram de ter relação direta com o *hardware* dos computadores, o que facilitou e agilizou enormemente o desenvolvimento e a manutenção de programas.

Década de 1960

Domínio da tecnologia física do estado sólido permitiu a integração de vários transistores em uma única embalagem com aproximadamente as mesmas dimensões de um único transistor.

Surgiram, então, os circuitos integrados, responsáveis pelo aparecimento dos computadores da 3ª geração.

1963 Burroughs lança o computador B-5000 com o sistema operacional *Master Control Program* (MCP), que oferecia multiprogramação, memória virtual com segmentação, multiprocessamento assimétrico, além de ser o primeiro sistema a ser desenvolvido em uma linguagem de alto nível

1964 IBM lança o *System*/360, que causou uma revolução na indústria de informática, pois introduziu o conceito de máquinas de porte diferente, porém com uma mesma arquitetura, permitindo a total compatibilidade entre os diferentes modelos.

1965 A Digital Equipment Corp. (DEC) lançou o PDP-8, representando a primeira linha de computadores de pequeno porte e baixo cuto.

1969 Ken Thompson, que trabalhou no projeto do sistema operacional MULTICS (*Multiplexed Information and Computing Service*), utilizou um PDP-7 para fazer sua própria versão de um sistema que viria a ser conhecido como UNIX.

Década de 1970

A integração em larga escala (*Large Scale Integration* – LSI) e a integração em muito larga escala (*Very Large Scale Integration* – VLSI), levaram adiante o projeto de miniaturização e barateamento dos equipamentos.

1971 A Intel Corp. produz seu primeiro microprocessador, o Intel 4004, e três anos depois o 8080, utilizado no primeiro microcomputador, o Altair.

Posteriormente a Zilog lançaria um processador concorrente ao da Intel, o Z80.

1975 Dennis Ritchie desenvolve a Linguagem C e, juntamente com Ken Thompson, portam o sistema UNIX para um PDP-11, concebido inicialmente em Assembly.

1976 Steve Jobs e Steve Wozniak produzem o Apple II de 8 bits, tornando-se um sucesso imediato.
Neste ano as empresas Apple e Microsoft são fundadas.

O sistema operacional dominante nos primeiros microcomputadores foi o CP/M (*Control Program Monitor*) da *Digital Research*.

O Cray-1 é lançado contendo 200.000 circuitos integrados e realizando 100 milhões de operações de ponto flutuante por segundo (100 MFLOPS).

As redes distribuídas difundiram-se, surgiram vários protocolos de rede, inclusive o NCP (precedessor do TCP/IP).

Surgem as primeiras redes locais.

Década de 1980

1981 A IBM entra no mercado de microcomputadores com o IBM PC (*Personal Computer*). O primeiro PC utilizava o processador Intel 8088 de 16 bits e o sistema operacional DOS (*Disk Operating System*) da Microsoft, muito semelhante ao CP/M.

1982 Fundada a Sun Microsystems, que passa a atuar fortemente no setor, lançando as primeiras estações RISC com o sistema operacional SunOS e, posteriormente, Sun Solaris.

A Universidade de Berkeley na Califórnia desenvolveu sua própria versão do sistema UNIX (*Berkeley Software Distribution* – BSD) e introduziu inúmeros melhoramentos, merecendo destaque o protocolo de rede TCP/IP (*Transmission Control Protocol / Internet Protocol*).

Com a evolução dos microprocessadores, principalmente da família Intel, surgem os primeiros sistemas operacionais comerciais que oferecem interface gráfica, como o Microsoft Windows e o OS/2.

O software de rede passa a estar fortemente relacionado com o sistema operacional e surgem os *sistemas operacionais de rede,* com destaque para o Novell Netware e o Microsoft LAN Manager.

Década de 1990

Grandes avanços de *hardware, software* e telecomunicações foram obtidos nesta década, conseqüência da evolução das aplicações que necessitavam cada vez mais de capacidade de processamento e armazenamento de dados, como em sistemas especialistas, sistemas multimídia, banco de dados distribuídos, inteligência artificial e redes neurais.

A evolução da microeletrônica permitiu o desenvolvimento de processadores e memórias cada vez mais velozes e baratos, além de dispositivos de E/S menores, mais rápidos e com maior capacidade de armazenamento.

Os componentes baseados em tecnologia VLSI evoluem rapidamente para o ULSI (*Ultra Large Scale Integration*).

Com o surgimento e a evolução da Internet, o protocolo TCP/IP passou a ser um padrão de mercado, obrigando os fabricantes a oferecer suporte a este protocolo.

Um fato importante nesta década foi o amadurecimento e popularização do software aberto (*open software*). Com a Internet, inúmeros produtos foram desenvolvidos e disponibilizados para uso gratuito, como sistemas operacionais (Linux e FreeBSD), banco de dados (MySQL), servidores Web (Apache), servidores de correio (Sendmail), dentre outros.

Década de 2000

Novas arquiteturas paralelas, baseadas em organizações de multiprocessadores não-convencionais, já se encontram em desenvolvimento em diversas universidades e centros de pesquisa.

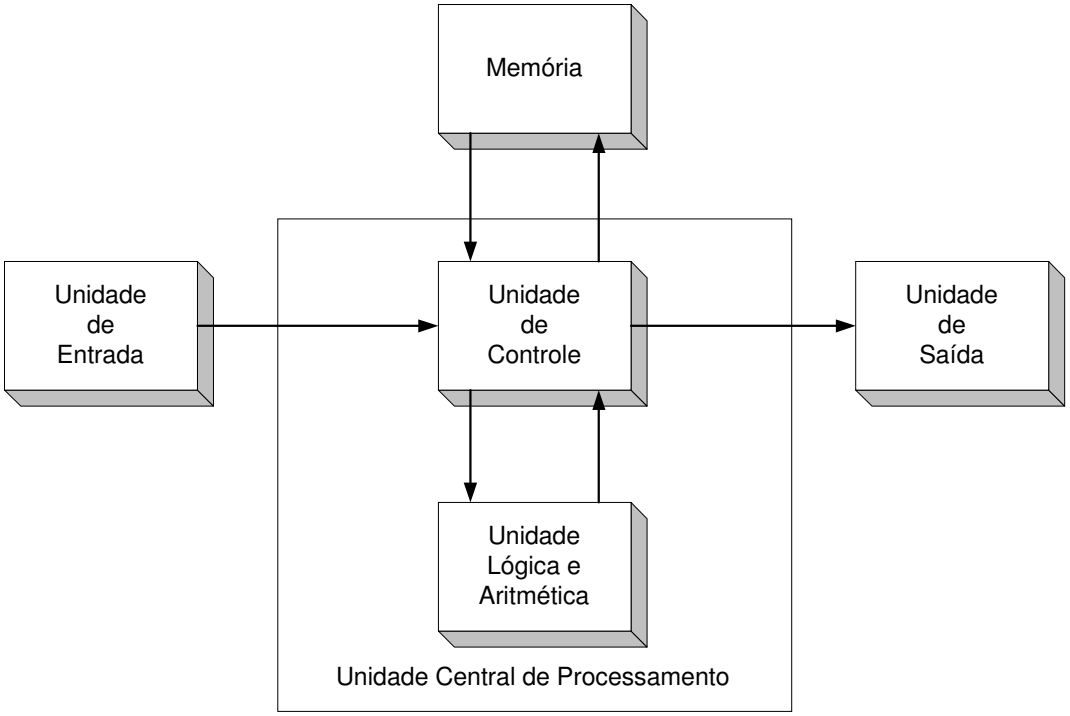
Novas formas de interação usuário-computador.

O conceito de processamento distribuído será mais explorado nos sistemas operacionais.

Evolução das redes sem fio (*wireless*).

1.3 Estrutura de um computador digital

O esquema da estrutura de um computador digital é dado pela figura:



Unidade de Entrada

Traduz informação de uma grande variedade de dispositivos em um código que a unidade central de processamento é capaz de entender.

Memória

Armazena os dados e os programas que “manipulam” estes dados.

Unidade Lógica e Aritmética

Nesta unidade são feitos todos os cálculos aritméticos e qualquer manipulação de dados, sejam eles numéricos ou não.

Unidade de Controle

É a unidade responsável pelo fluxo dos dados. Ela obtém dados armazenados na memória e interpreta-os. Controla a transferência de dados da memória para a unidade lógica e aritmética, da entrada para a memória e da memória para a saída.

Unidade de Saída

Os dados processados são convertidos, por esta unidade, de impulsos elétricos em palavras ou números que podem ser "escritos" em impressoras ou "mostrados" em vídeos ou numa séria de outros dispositivos.

1.4 Algoritmo

Conceituação

Para estabelecer o conceito de algoritmo, que é da maior importância em Ciência da Computação, será antes fixado o conceito de ação.

Ação é um **acontecimento** que, a partir de um **estado inicial**, após um período de tempo **finito**, produz um **estado final previsível** e bem **definido**.

Adota-se a seguinte definição de **algoritmo**:

*"**Algoritmo** é a descrição de um conjunto de comandos que, obedecidos, resultam numa sucessão finita de ações".*

Geralmente, um algoritmo se destina a resolver um problema: fixa um padrão de comportamento a ser seguido, uma norma de execução a ser trilhada, para se atingir, como resultado final, a solução de um problema.

Exemplo:

Descrever um algoritmo que mostre todos os números da série de Fibonacci menores que um valor dado *n*.

```
Algoritmo Fibonacci
inicio
  Escreva os termos de Fibonacci inferiores a n
fim
```

Refinamentos

Um algoritmo é considerado **completo** se os seus comandos forem do entendimento do seu destinatário.

Num algoritmo, um comando que não for do entendimento do destinatário terá de ser desdobrado em novos comandos, que constituirão um **refinamento** do comando inicial.

Se um algoritmo é formado não apenas por um comando, mas por vários, isto significa que na sua execução não se consideram apenas o **estado inicial** e o **final** de uma ação dele resultante, mas que se consideram também **estados intermediários** que delimitam as ações decorrentes de cada comando.

Exemplo:

O algoritmo dos termos de Fibonacci poderia ser refinado em:

```
Refinamento Escreva os termos de Fibonacci inferiores a n
inicio
  Receba o valor de n
  Processe os 2 primeiros termos
  Processe os termos restantes
fim
```

Um algoritmo e os seus refinamentos são formados por **comandos**, que determinam as ações a serem executadas pelo seu destinatário e por **estruturas de controle** que determinam a ordem em que os comandos devem ser executados, se devem ser executados ou não e quando devem ser repetidos.

No exemplo anterior vigora a estrutura de controle mais simples: **estrutura seqüencial**, segundo a qual os comandos devem ser executados um após o outro, na mesma ordem em que aparecem escritos.

Se um comando de um refinamento for um tanto vago, ele poderá, por sua vez, ser desdobrado em novos comandos, produzindo-se o refinamento de um refinamento, e assim sucessivamente.

Exemplo:

O comando "`Processe os 2 primeiros termos`" pode ser desdobrado em:

```
Refinamento Processe os 2 primeiros termos
inicio
  Atribua o valor 1 ao primeiro termo
  se ele for menor do que n
     então escreva-o
  fim se
  Atribua o valor 1 ao segundo termo
```

```
  se ele for menor do que n
     então escreva-o
  fim se
fim
```

Neste refinamento, aparece uma segunda estrutura de controle: a **estrutura condicional**:

se condição
  então comandos
fim se

O comando "`escreva-o`" só será executado se a condição "`ele for menor do que` *n*" for verdadeira. O comando "`escreva-o`" não será executado se a condição for falsa, isto é, se ele for maior ou igual a *n*.

Vê-se, portanto, que um algoritmo, através de estruturas condicionais, pode provocar ou não a realização de uma ação.

Uma terceira estrutura de controle, a **estrutura de repetição**, será necessária ao se desdobrar o comando "`processe os termos restantes`".

Exemplo:

Refinamento de "`Processe os termos restantes`":

```
Refinamento Processe os termos restantes
inicio
  repita
    Calcule novo termo somando os 2 anteriores
    se novo termo for maior ou igual a n
     então interrompa
    fim se
    Escreva novo termo
  fim repita
fim
```

Na estrutura de repetição, os comandos e as estruturas de controle abrangidos devem ser executados repetidamente até que se verifique uma condição (no caso, o novo termo a ser maior ou igual a *n*) para que se **interrompa** a repetição.

Após esses **refinamentos sucessivos**, o algoritmo pode ser considerado completo, a menos que o destinatário não saiba fazer a adição de dois termos ou não seja capaz de entender diretamente algum comando.

Exemplo:

Algoritmo completo para escrita dos termos de Fibonacci inferiores a *n*:

```
Algoritmo Fibonacci
inicio
  Receba o valor de n
    {Processamento dos 2 primeiros termos}
  Atribua o valor 1 ao primeiro termo
  se ele for menor do que n
     então escreva-o
  fim se
  Atribua o valor 1 ao segundo termo
  se ele for menor do que n
     então escreva-o
  fim se
    {Processamento dos termos restantes}
  repita
    Calcule novo termo somando os 2 anteriores
    se novo termo for maior ou igual a n
     então interrompa
    fim se
    Escreva novo termo
  fim repita
fim
```

1.5 Algoritmos Estruturados

Com a evolução, os problemas levados aos computadores são cada vez de maior porte e maior complexidade. Os algoritmos para resolvê-los ainda devem ser desenvolvidos pelo ser humano, mas podem ultrapassar os limites de sua compreensão.

Por esta razão, nas últimas décadas surgiram técnicas que permitem sistematizar e ajudar o desenvolvimento de algoritmos para a resolução de grandes e complexos problemas nos computadores: são as técnicas de **desenvolvimento estruturado** de algoritmos.

Os objetivos destas técnicas são:

- Facilitar o desenvolvimento dos algoritmos;
- Facilitar o seu entendimento;
- Antecipar a comprovação da sua correção;
- Permitir que o seu desenvolvimento possa ser empreendido simultaneamente por uma equipe de pessoas.

Para atingir estes objetivos, o desenvolvimento estruturado preconiza que:

- Os algoritmos sejam desenvolvidos por **refinamentos sucessivos** partindo de uma descrição geral e, gradativa e sucessivamente, atacando as minúcias e particularidades. Este desenvolvimento também se denomina "construção hierárquica de algoritmos" e "desenvolvimento *top/down*".
- Os sucessivos refinamentos são módulos, que delimitam poucas funções e são os mais independentes possíveis, isto é, conservam poucos vínculos com outros módulos.
- Nos módulos deve ser usado um número **limitado** de diferentes **comandos** e de diferentes **estruturas de controle**.

O refinamento sucessivo dos algoritmos permite uma abordagem mais segura e objetiva do problema. A integração dos módulos é mais perfeita, porque cada módulo é atacado quando já se estudou claramente o ambiente em que ele deve atuar.

A divisão em módulos funcionais permite contornar a limitação humana para compreender a complexidade. Cada módulo pode ser desenvolvido ou analisado de forma quase independente, dados os poucos vínculos que deve manter com os outros módulos do algoritmo.

1.6 Linguagens de Programação

Para armazenar um algoritmo na memória de um computador e para que ele possa, em seguida, comandar as operações a serem executadas, é necessário que ele seja **programado**, isto é, que seja transcrito para uma linguagem que o computador possa "enteder", direta ou indiretamente.

Os computadores só podem executar **diretamente** os algoritmos expressos em **linguagem de máquina**, que é um conjunto de instruções capazes de ativar diretamente os dispositivos eletrônicos do computador.

A linguagem de máquina tem vários inconvenientes para os humanos. É diferente para cada tipo de computador, pois depende de sua arquitetura. Além disto, é extremamente rudimentar e exige que, mesmo as operações mais simples ainda sejam refinadas, para expressá-las em termos de registros, acumuladores e outros dispositivos da máquina.

Além disso, a linguagem de máquina é expressa em forma numérica (binária ou hexadecimal), que a torna pouco expressiva para os seres humanos.

Nos primeiros computadores, a linguagem de máquina era a única opção para fazer a programação.

Logo surgiu a idéia de se escreverem programas em **Linguagem simbólica**, mais conhecida por **Linguagem Assembly** ou **Linguagem montadora**.

Na Linguagem Assembly as instruções não são expressas apenas por números, mas também por letras e símbolos mais significativos para os humanos. O posicionamento dos dados e instruções na memória é também feito de forma simbólica.

Um programa em Assembly, para controlar o computador, deve primeiro ser transformado em linguagem de máquina. Como cada comando da linguagem Assembly corresponde a um comando em linguagem de máquina, esta transformação pode ser feita facilmente pelo próprio computador, através de um programa chamado **montador** ou **Assembler**.

O sucesso da linguagem Assembly animou os primeiros pesquisadores a criar linguagens em que a programação era feita através de uma notação matemática e de algumas palavras da língua inglesa, deixando ao próprio computador a tarefa de traduzir este programa para a linguagem de máquina, através de um programa chamado **compilador**.

A primeira destas linguagens, que teve ampla aceitação, surgiu em 1957 e é ainda hoje utilizada. Trata-se da linguagem FORTRAN, nome formado com partes das palavras “*Fom*ula *Tran*slation”.

Além da grande facilidade, uma imensa vantagem de se escrever os programas em **linguagens de alto nível**, como passaram a ser chamadas a linguagem FORTRAN e outras que se seguiram, é a sua quase total independência da máquina a ser usada.

O programador não precisa se deter em particularidades do computador que irá usar. Um programa escrito em linguagem de alto nível, geralmente com poucas alterações, é aceito por qualquer computador.

A escolha da linguagem de programação para se usar em um computador depende, antes de tudo, da existência de um programa **compilador** (que traduza o algoritmo escrito na linguagem escolhida para a linguagem de máquina) ou de um programa **interpretador** (que interprete cada comando do programa e execute uma série de instruções que a ele correspondem).

Existindo compiladores ou interpretadores para diversas linguagens, a escolha pode ser pela linguagem preferida ou mais familiar para o programador, por aquela melhor implementada no computador, por aquela mais adequada para o tipo de aplicação que se deseja fazer etc.

Mas, para se resolver um problema num computador, mais importante que a escolha da linguagem de programação é o desenvolvimento de um algoritmo adequado.

O algoritmo deve ser desenvolvido objetivando sobretudo a clareza, permitindo que os erros cometidos sejam detectados o quanto antes, evitando excessivas revisões e visando facilitar futuras modificações.

Para se conseguir isto de uma forma natural e eficiente, deve-se adotar técnicas de desenvolvimento estruturado dos algoritmos. Depois disto, a programação consistirá quase que só numa transcrição do algoritmo obtido.

Bibliografia

FARRER, H. et. alli. *Algoritmos Estruturados.* 3ª. edição. Rio de Janeiro. LTC – Livros Técnicos e Científicos Editora S/A, 1999.

GUIMARÃES, A. M. e LAGES, N. A. C. *Algoritmos e Estrutura de Dados.* 1ª. edição. Rio de Janeiro. LTC – Livros Técnicos e Científicos Editora S/A, 1985.

MACHADO, F. B. e MAIA, L. P. *Arquitetura de Sistemas Operacionais.* 3ª. edição. Rio de Janeiro. LTC – Livros Técnicos e Científicos Editora S/A, 2002.