

PCI- INTRODUÇÃO

Profa. Mercedes Gonzales Márquez

De algoritmos a programas

- Um algoritmo descreve uma sequência de comandos e passos que devem ser executados para resolver um problema.
- Um algoritmo não necessariamente é executado por uma máquina. Pode ser executado pelo homem, exemplo: uma receita de bolo, a troca de um pneu, etc.
- Para o algoritmo ser executado pelo computador, este deve ser codificado em uma linguagem de programação.

De algoritmos a programas

- Um algoritmo pode ser descrito de várias formas, dentre elas, em português (linguagem natural) ou em portugol (linguagem algorítmica em português próxima de uma linguagem de programação).
- A vantagem de se implementar um algoritmo em uma linguagem de programação é que geramos um programa que será executado pelo computador para resolver o problema.

De algoritmos a programas

- Nesta disciplina estudaremos como usar um computador para resolver problemas.
 - (a) Definiremos um problema a ser resolvido,
 - (b) descreveremos uma solução (algoritmo) e
 - (c) implementaremos esta solução criando um programa.
- Nesta disciplina usaremos a linguagem C para implementar nossos algoritmos.

De algoritmos a programas

- Esta disciplina estará fortemente ligada à disciplina Algoritmos e Estruturas de Dados I (AEDI), a qual explanará com detalhes o processo de construção de algoritmos.
- Recomenda-se fortemente que você estude esta disciplina em conjunto com AEDI.
- Vejamos três exemplos de algoritmos nas suas três representações:
 1. Linguagem natural
 2. Linguagem algorítmica (Portugol)
 3. Linguagem de programação C

De algoritmos a programas

Problema 1. Encontre a média aritmética de 3 notas (N1, N2 e N3) de um aluno.

A) Algoritmo na linguagem natural

1. Forneça os valores das notas N1, N2 e N3.
2. Calcule a média aritmética $media = (N1 + N2 + N3) / 3$
3. Imprima a média na tela.

De algoritmos a programas

Problema 1. Encontrar a média aritmética de 3 notas (N1, N2 e N3) de um aluno.

B) Algoritmo em Portugol

1. Leia N1, N2 e N3.
2. $media \leftarrow (N1+N2+N3)/3$
3. Escreva media

De algoritmos a programas

Problema 1. Encontrar a média aritmética de 3 notas (N1, N2 e N3) de um aluno.

C) Programa em C

```
#include <stdio.h>
int main(){
    float N1, N2, N3, media; /* declaracao de variaveis */
    printf ("Leitura das tres notas\n");
    printf ("Leitura da primeira nota\n");
    scanf ("%f",&N1);
    printf ("Leitura da segunda nota\n");
    scanf ("%f", &N2);
    printf ("Leitura da terceira nota\n");
    scanf ("%f",&N3);
    media=(N1+N2+N3)/3;
    printf ("A media eh:");
    printf ("%f",media);
}
```

De algoritmos a programas

Problema 2. Encontre a média aritmética de 3 notas (N1, N2 e N3) de um aluno e informe se ele aprovou ou não.

A) Algoritmo na linguagem natural

1. Forneça os valores das notas N1, N2 e N3.
2. Calcule a média aritmética $media = (N1 + N2 + N3) / 3$
3. Se media for pelo menos 6 imprima “Aprovado”, caso contrário imprima “Reprovado”.

De algoritmos a programas

Problema 2. Encontre a média aritmética de 3 notas (N1, N2 e N3) de um aluno e informe se ele aprovou ou não.

B) Algoritmo em Portugol

1. Leia N1, N2 e N3.
2. $media \leftarrow (N1 + N2 + N3) / 3$
3. Se $(media \geq 6)$ então
 escreva “Aprovado”
Senão
 escreva “Reprovado”

De algoritmos a programas

Problema 2. Encontre a média aritmética de 3 notas (N1, N2 e N3) de um aluno e informe se ele aprovou ou não.

C) Programa em C

```
float N1, N2, N3, media; /* declaracao de variaveis */
printf ("Leitura das tres notas\n");
printf ("Leitura da primeira nota\n");
scanf ("%f",&N1);
printf ("Leitura da segunda nota\n");
scanf ("%f", &N2);
printf ("Leitura da terceira nota\n");
scanf ("%f",&N3);
media=(N1+N2+N3)/3;
printf ("A media eh:");
printf ("%f",media);
if (media>=6)
    printf ("\nAluno aprovado");
else
    printf ("\nAluno reprovado");
```

De algoritmos a programas

Problema 3. Encontre as raízes x_1 e x_2 de uma equação quadrática $ax^2 + bx + c = 0$

A) Algoritmo na linguagem natural

1. Forneça os valores de a , b e c .
2. Calcule o discriminante $d = b^2 - 4ac$
3. Se d for positivo calcule as duas raízes seguintes:
 $x_1 = (-b + \sqrt{d})/2a$ e $x_2 = (-b - \sqrt{d})/2a$

De algoritmos a programas

Problema 3. Encontre as raízes x_1 e x_2 de uma equação quadrática $ax^2 + bx + c = 0$

B) Algoritmo em Portugol

1. Leia (a,b,c)
2. $d \leftarrow b*b - 4*a*c$
3. Se ($d \geq 0$)
 - $x_1 \leftarrow (-b + \text{sqrt}(d)) / 2*a$
 - $x_2 \leftarrow (-b - \text{sqrt}(d)) / 2*a$

De algoritmos a programas

- Exemplo de solução:
 1. Sejam os valores $a=1, b=0$ e $c=-1$
 2. $d= b^2 - 4c$, teremos $d = 4$.
 3. Como $d>0$, calculamos $x_1 = (-b + \sqrt{d})/2a = 1$ e $x_2 = (-b - \sqrt{d})/2a = -1$

De algoritmos a programas

Problema 3. Encontre as raízes x_1 e x_2 de uma equação quadrática $ax^2 + bx + c = 0$

C) Programa em C

```
float a,b,c,d;
printf ("Leitura dos tres coeficientes\n");
scanf ("%f %f %f",&a,&b, &c);
d=b*b-4*a*c; /* discriminante*/
if (d>0){
    x1=(-b+sqrt(d))/2*a;
    x2 =(-b-sqrt(d))/2*a;
    printf ("As raizes sao:");
    printf ("%f %f",x1,x2);
}
```

Conceitos básicos

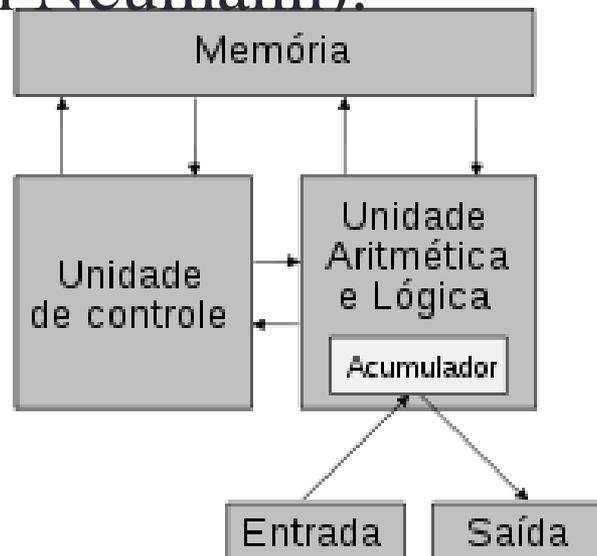
A seguir daremos alguns conceitos básicos de computadores para contextualizar melhor o conteúdo da nossa disciplina.

Computador

- Um computador é uma máquina que, a partir de uma entrada, realiza um número muito grande de cálculos matemáticos e lógicos, gerando uma saída.
- O conjunto de componentes que formam computador se dividem em duas partes principais: **Hardware e Software.**

Hardware

- É a parte mecânica e física da máquina, com seus componentes eletrônicos e peças.
- Estes dispositivos seguem uma organização básica (Arq. de Von Neumann).



Hardware

- Todo o hardware opera com sinais digitais: sem energia e com energia.
- Chamamos estes sinais de Bit :Valores 0 ou 1. Byte : um agrupamento de 8 bits.
- Todas as informações armazenadas no computador são representadas por números 0s e 1s. Informações como letras, símbolos, imagens, programas são todas vários 0s e 1s.

Hardware

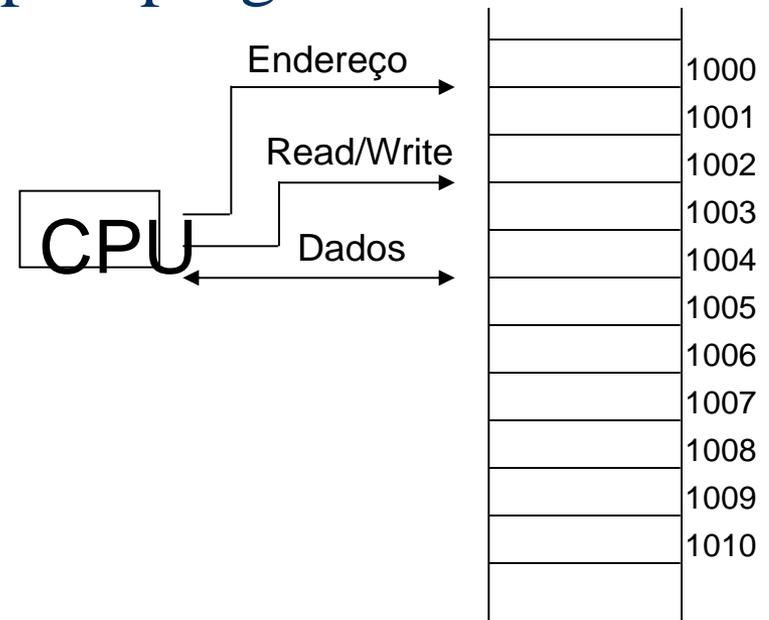
- **Unidade de entrada** – Traduz informação de um dispositivo de entrada em um código que a CPU entende (padrões de pulsos elétricos compreensíveis ao computador).
- **Unidade de saída** – converte os dados processados, de pulsos elétricos em palavras ou números que podem ser escritos em vídeos ou outros dispositivos de saída.

Hardware

- **Memória** –Número finito de localizações que são identificadas por meio de um único endereço. Armazena os dados e o próprio programa.

Escrita – CPU envia endereço da posição de memória a ser escrita e dados a escrever.

Leitura – CPU envia endereço da posição de memória a ser lida e recebe dados.



Hardware

- **Unidade lógica e aritmética** – São executadas operações matemáticas de adição, multiplicação e divisão e operações lógicas como conjunção, disjunção, ou exclusivo e outras.
- **Unidade de controle** – Responsável pelo “tráfego” de dados. Controla a transferência de dados da memória para a unidade lógica e aritmética, da entrada para a memória e da memória para a saída.

Software

São conjuntos de procedimentos básicos que fazem que o computador seja útil executando alguma função. A essas “ordens” preestabelecidas chamamos também de programas.

Linguagem de Programação: Consiste da sintaxe (gramática) e semântica (significado) utilizada para escrever (ou codificar) um programa.

- Alto nível:

Software

- Alto nível: Linguagem de codificação de programa independente do tipo de máquina e de fácil utilização pelo ser humano. Ex.Pascal, C, Java, Python etc.
- Baixo nível: Linguagem de codificação baseada em mnemônicos. Dependente do tipo de máquina e de fácil tradução para a máquina. Conhecida como linguagem assembly.
- Linguagem de Máquina. Conjunto de códigos binários que são compreendidos pela CPU de um dado computador. Depende do tipo de máquina.

Organização Básica de um ambiente computacional

Um ambiente computacional é organizado como uma hierarquia de funções, onde cada uma é responsável por uma tarefa específica.

Programas de Aplicação
Compiladores/Interpretadores
Sistema operacional
Hardware

Organização Básica de um ambiente computacional

- Programas de Aplicação. Nesta disciplina nos propomos criar novos programas de aplicação. Os programas de aplicação usam compiladores ou interpretadores como tradutores de uma linguagem abstrata para linguagem de máquina.
- Nesta disciplina usaremos um compilador para uma linguagem de programação C.

Organização Básica de um ambiente computacional

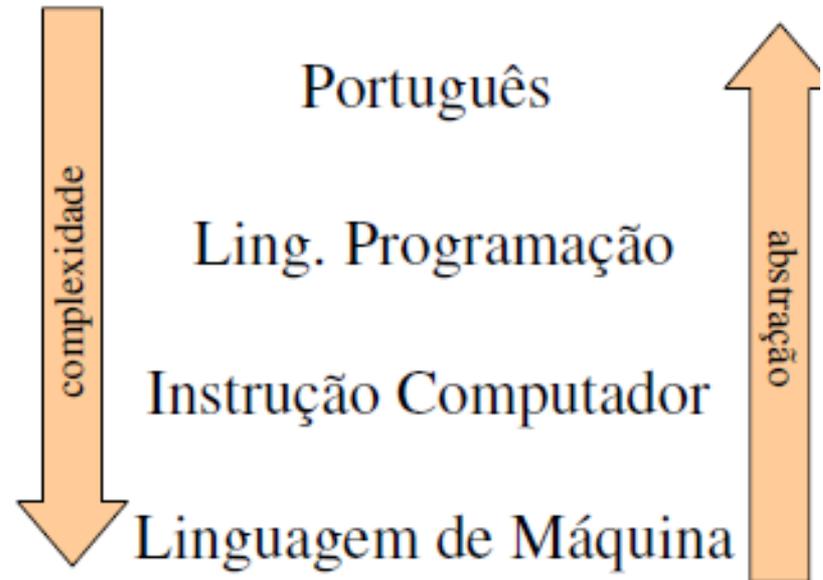
- **Compilador:** Traduz programas codificados em linguagem de alto ou baixo nível (i.e. código fonte) para linguagem de máquina (i.e. código executável).
Ex: O assembler transforma um programa em assembly para linguagem de máquina. Uma vez compilado o programa pode ser executado em qualquer máquina com o mesmo sistema operacional para o qual o programa foi compilado.

Organização Básica de um ambiente computacional

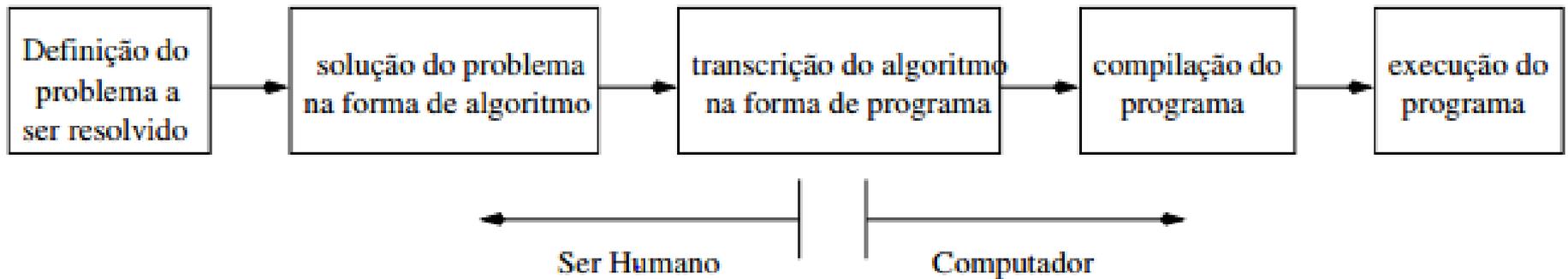
- Interpretador: Traduz o código fonte para o código de máquina diretamente em tempo de execução. Exemplos: Python, Tcl/Tk, LISP.
- Sistema Operacional: Conjunto de programas que gerenciam e alocam recursos de hardware e software. Ex. Linux, Unix, MAC OS, Windows, etc.

Níveis de abstração

- Português:
 - Fácil, intuitivo
 - Computador não entende
 - Ambíguo, mal definido
- Linguagem de Máquina:
 - Complexo e trabalhoso
 - Única forma aceita pelo computador
 - Preciso, bem definido
 - Envolve detalhes específicos do computador, irrelevantes para o algoritmo



Etapas da resolução de problemas usando o computador



Da apostila do Prof.
Alexandre Xavier Falcão

Compilando um programa no CodeBlocks do windows

1. Crie um arquivo novo utilizando File → New → Empty file.
2. Salve o arquivo após digitar o cabeçalho, utilizando um nome apropriado terminado com a extensão .c . Para tanto, vá em File → Save File (CTRL + S) e escolha um nome. No caso de exemplo utilizamos oi.c.

Compilando um programa no CodeBlocks do windows

3. Insira o código lembrando sempre de indentá-lo com a tecla TAB e comentá-lo apropriadamente. Note que o Code::Blocks tende a indentar o código automaticamente, mas haverá ocasiões em que a indentação deverá ser corrigida manualmente.
- Após alterar o arquivo .c, salve-o e aperte a tecla F9 (Build → Build and Run) para compilá-lo e executá-lo. Pressione também F2 para aparecer a um retângulo abaixo do código com a saída da compilação caso já não esteja aparecendo.

Primeiro Programa

```
/* file: oi.c
 * Este programa escreve a mensagem "Oi pessoal"
 * na tela. O programa foi tomado do livro
 * "Art and Science of C" de Eric Roberts. */
#include <stdio.h>
int main(){
    printf("Oi pessoal.\n");
}
```

Primeiro Programa

- **Estrutura do código fonte**
 - Comentários
 - Diretivas de compilador
 - Procedimento principal

Primeiro Programa

Comentários

```
/* file: oi.c  
* Este programa escreve a mensagem "Oi  
* pessoal" na tela. O programa foi tomado do  
* livro "Art and Science of C" de Eric Roberts. */
```

Diretivas de
Compilador

```
#include <stdio.h>
```

Procedimento
Principal

```
int main()  
{  
    printf("Oi pessoal.\n");  
}
```

Primeiro Programa

Comentários

- Texto ignorado pelo compilador
- Documentação útil para descrever trechos do algoritmo
- Possível em qualquer posição do código fonte
- Duas formas para comentários:

- Uma linha:

```
// Comentário ...
```

- Várias linhas:

```
/* Comentário...
```

```
mais comentários ... */
```

Primeiro Programa

Diretivas de Compilador:

- Informam outros arquivos que devem ser consultados antes de compilar.
- Definem parâmetros utilizados pelo compilador.
- São colocadas no início do código fonte.

Exemplos:

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
#include <math.h>
```

Primeiro Programa

Procedimento principal:

- Sequência de instruções
- Pontuação: ponto-e-vírgula termina instruções
- Chaves agrupam instruções relacionadas

```
int main(){  
    printf("Oi pessoal.\n");  
}
```

Primeiro Programa

- Caso o programa não esteja de acordo com as regras da linguagem, erros de compilação ocorrerão. Ler e entender estes erros é muito importante.

```
#include <stdio.h>
int main() {
    printf("Oi pessoal!\n");
```

Após a compilação teremos oi.c: In function `main':
oi.c:5: error: syntax error at end of input