

Algoritmos e Lógica de Programação II



Unidade 01

Introdução a Linguagem C

Prof. Rogério Napoleão Júnior



Introdução

- Algoritmo na Computação
 - Corresponde a transformar um conjunto de dados de **entrada** em um conjunto de dados de **saída** por meio de **instruções** computacionais.
- Linguagem C
 - Linguagem de programação de propósito geral
 - Sintaxe muito parecida com outras linguagens: Pascal, Delphi, Java, C++, C#, etc.
- Não depende de um hardware específico

Introdução

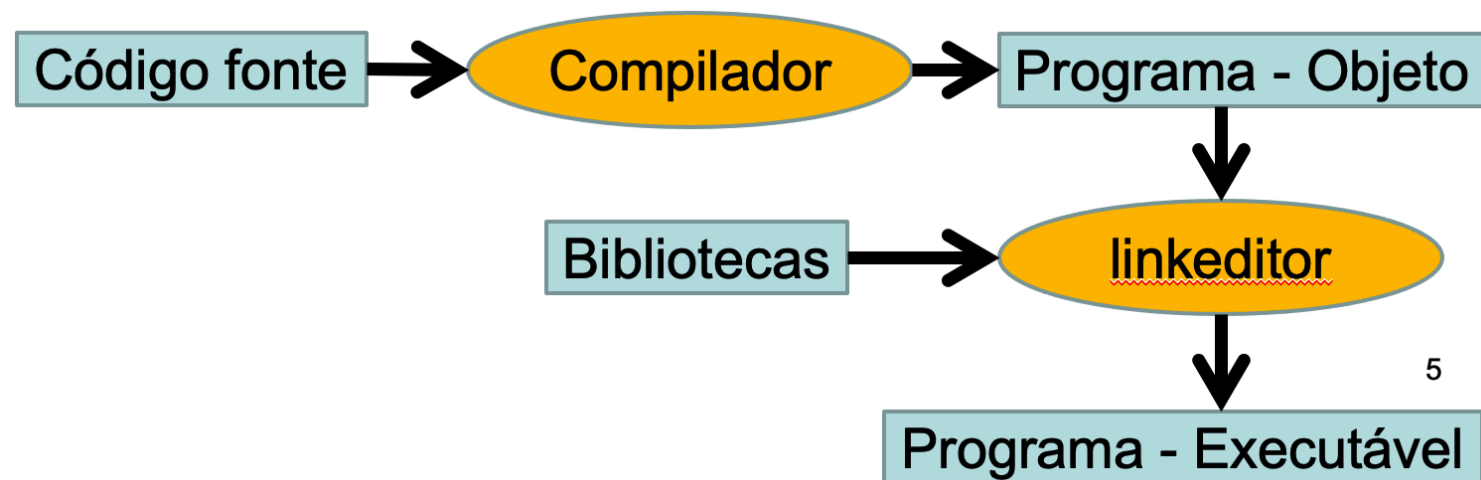
- Combina vantagens de uma linguagem de alto nível com a eficiência das linguagens de máquina (*assembly*)
- Compiladores e Ambientes de Prog. (IDEs)
 - GCC
 - Dev C++
 - Visual Studio
 - Turbo C
 - Visual C



Introdução

- Linguagens:
 - Interpretadas
 - Compiladas (C)

Processo de compilação + linkedição



Introdução

- Estrutura básica de um programa em C
 - Inclusões de bibliotecas
 - Corpo principal do código - função main
 - Comandos geralmente finalizam com “;”

- Exemplo:

```
1 #include <stdio.h>
2 int main(){
3     printf("Olá, mundo!");
4     return 0;
5 }
```

Bibliotecas

- Conjuntos de recursos prontos disponíveis para utilização do programador
- São inseridas no cabeçalho do código, após compilado é vinculado ao programa pela etapa de linkedição
- `stdio.h`
 - Biblioteca contendo funções para entrada e saída de dados
 - Exemplo – `printf("Olá mundo");`

Comentários

- Há situações em que o programador quer deixar anotações registradas no código fonte
 - Tais informações não podem ser interpretadas como instruções executáveis
 - São apenas informações instrutivas (documentação)
 - O compilador deve desconsiderar qualquer coisa que esteja em comentário

Comandos “barra-barra” & “barra-asterisco”:

// <comentário>

/* <comentário> */

Identificadores

- Existem **comandos** com identificadores fixos:
 - Todos escritos em letras minúsculas
 - Não podem ser substituídos (sobrescritos)
 - Não podem ser “reutilizados”
- É comum **dar nomes** a diferentes estruturas dentro de um programa
 - Variáveis
 - Constantes
 - Tipos de dados
 - Funções
 - Rótulos

Identificadores

- **Regras para a criação de identificadores** (nomeação de estruturas)
 - Nunca começam com um número
 - Geralmente começam com uma letra ou *underline*
 - Podem conter letras, *underlines* e números
 - O primeiro caractere não pode ser um número
 - Não admitem acentos, espaços e nem caracteres especiais
- Observação importante: a linguagem C é **case sensitive** – maiúsculo ≠ minúsculo

Identificadores

Identificadores válidos	Identificadores inválidos
A	2 ^a
a	b@
media	media idade
altura2	x*y
media_idade	#media
x36	idade!

Elementos de um Programa

- Tipos de dados
- Palavras reservadas
- Constantes
- Variáveis
- Atribuição
- Operadores
 - Aritméticos, relacionais e lógicos
- Funções
- Entrada e saída de dados

Tipos de Dados

- Oito tipos primitivos:
 - **char**: caracteres simples e strings (cadeias de caracteres)
 - **int**: dados numéricos sem casas decimais
 - **float**: números com casas decimais (valores em ponto flutuante)
 - **double**: ponto flutuante com precisão dupla
 - **bool**: valor lógico (verdadeiro ou falso)
 - **enum**: dados enumerados
 - **void**: ausência de valores, ocupam 0 bits em memória
 - **pointer**: localização de memória

Tipos de Dados

- **Modificadores de tipos de dados:**
 - **unsigned:** utilizado para declarar que um valor numérico não terá sinal
 - Duplica a faixa de valores
 - **short:** reduz a capacidade de armazenamento
 - **long:** armazena a capacidade de armazenamento
- Os modificadores podem ser utilizados em conjunto com os tipos. Por exemplo:
 - long int
 - unsigned float

Palavras Reservadas

- Comandos específicos que têm significado próprio:
 - int, float, double, void, bool, char, enum
 - short, long, unsigned
 - if, else, do, while, for, switch, case, break, default
 - return
 - typedef, struct, union

Variáveis

- Permitem armazenar e acessar uma informação
 - Valores armazenados em memória
- Permitem que valores sejam alterados conforme a necessidade ao longo do programa
- Toda variável possui:
 - UM TIPO
 - UM NOME (identificador)

Variáveis

- *Sintaxe de criação de variáveis:*
 - `<tipo> <identificador>;`
- Pode-se declarar mais de uma variável na mesma linha
- Exemplos:

```
1 float salario;  
2 int idade, ano;  
3 char nome[20];  
4 bool brasileiro;
```

Constantes

- Alguns valores são fixos durante todo o programa
- As informações contidas em constantes não variam ao longo do programa
- *Sintaxe:*
 - `#define <identificador> <valor>`
- Exemplos:

```
1 #define PI 3.1416
2 #define MSG_ERRO "Erro!"
```

Atribuição

- A **atribuição** é uma operação para armazenar ou alterar o conteúdo de uma variável.
 - Pode-se atribuir (dependendo do tipo da variável):
 - Valores literais
 - Valores constantes
 - Outras variáveis
 - Resultados de expressões
- **Em linguagem C, usa-se o símbolo “igual”:**
<variável> = <valor>;

Atribuição

- Exemplo

```
1  idade = 42;  
2  ano = 2010;  
3  ano_atual = ano + idade;  
4  custo = 66.89;  
5  juros = custo * 1.05;  
6  brasileiro = true;
```

Operadores Aritméticos

- Soma
 - Símbolo: +
- Subtração
 - Símbolo: -
- Multiplicação
 - Símbolo: *
- Divisão
 - Símbolo: /
- Módulo
 - Resto de divisão inteira
 - Símbolo: %
- Prioridades – parênteses
 - Símbolo: ()

Operadores Aritméticos

- Ejemplos

```
1  int A = 2, B = 3, Resultado = 0;  
2  
3  Resultado = A + B;  
4  Resultado = B - A;  
5  Resultado = A * B;  
6  Resultado = (A + 4) / 3;
```

Operadores Relacionais

- Maior que
 - Símbolo: $>$
- Maior ou igual a
 - Símbolo: \leq
- Menor que
 - Símbolo: $<$
- Menor ou igual a
 - Símbolo: \geq
- Igual
 - Símbolo: $==$
- Diferente
 - Símbolo: \neq

Operadores Relacionais

Exemplos

```
1  if (A < B)
2
3  if (A >= 3.14)
4
5  if (B != 10)
6
7  if ((A - B) == 0)
```


Operadores Relacionais

- Negação
 - Inverte um valor lógico
 - Símbolo: !
- Conjunção
 - Operação “E” lógica
 - Símbolo: &&
- Disjunção
 - Operação “OU” lógica
 - Símbolo: ||

Operadores Relacionais

Exemplos

```
1  if (A < 10 && A > 0)
```

```
2
```

```
3  if (A >= 0 || B >= 0)
```

```
4
```

```
5  if (! (A == B))
```

Funções Intrínsecas

- Existem operações complexas que já foram desenvolvidas em forma de funções
 - É preciso identificar a biblioteca à qual uma função intrínseca pertence
- Exemplos:
 - `ceil(x)`
 - `abs(x)`
 - `floor(x)`
 - `log(x)`
 - `log10(x)`
 - `z = modf(x, &y)`
 - `pow(x,y)`
 - `sqrt(x)`
 - `printf("texto")`
 - `scanf("%d", &x)`

Funções Intrínsecas

- Exemplos:

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <math.h>
4  int main(){
5      float A = 4, Resultado;
6
7      Resultado = sqrt(A);
8
9  }
```

