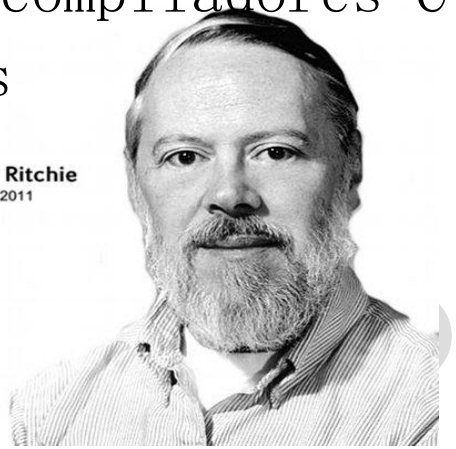
Introdução a Linguagem C

Laboratório de Programação

Prof.: André Bezerra

Histórico

- Criada em 1972 por Dennis Ricthie no centro de Pesquisas da Bell Laboratories. Sua primeira utilização importante foi a reescrita do Sistema Operacional UNIX, que até então era escrito em ASSEMBLY.
- A simplicidade de sua implementação permitiu a extensão da linguagem e a criação de compiladores C para praticamente todas as plataformas de hardware e sistemas operacionais.



Aplicações escritas em C

- Sistema Operacional: UNIX
- Planilhas: 1,2,3 e Excel
- Banco de Dados: dBase III, IV e Access (gerenciador de base de dados).
- Aplicações Gráficas: Efeitos Especiais de filmes como Star Trek e Star War.

Características da Linguagem C

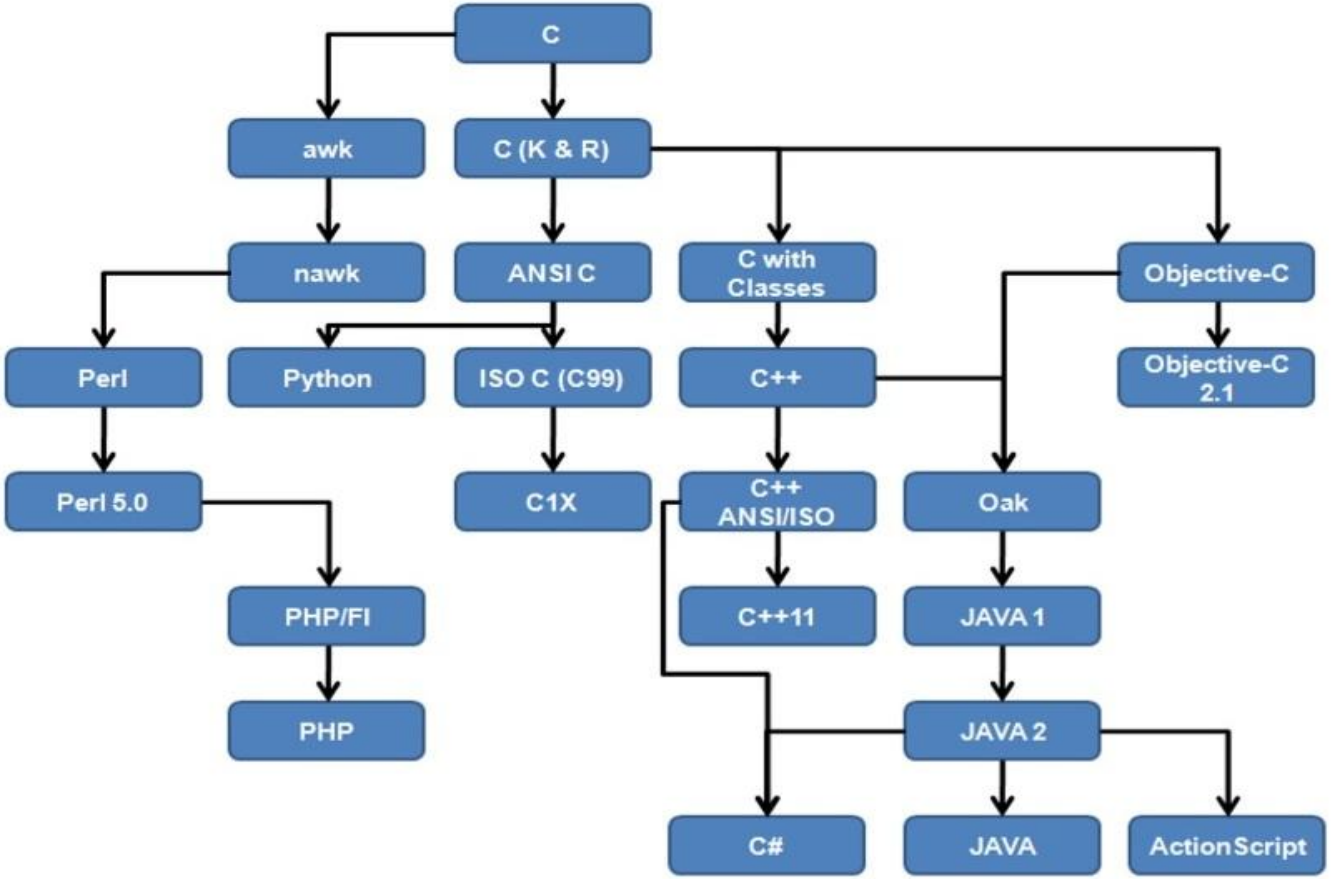
- Linguagem de propósito geral
  - Usada para escrever processadores de texto, planilhas, sistemas operacionais, programas de comunicação, programas para automação industrial, SGBDs, navegadores e servidores Web, etc.
- Possui características de alto e de baixo nível
- Excelente performance
- Muito popular

Características da Linguagem C

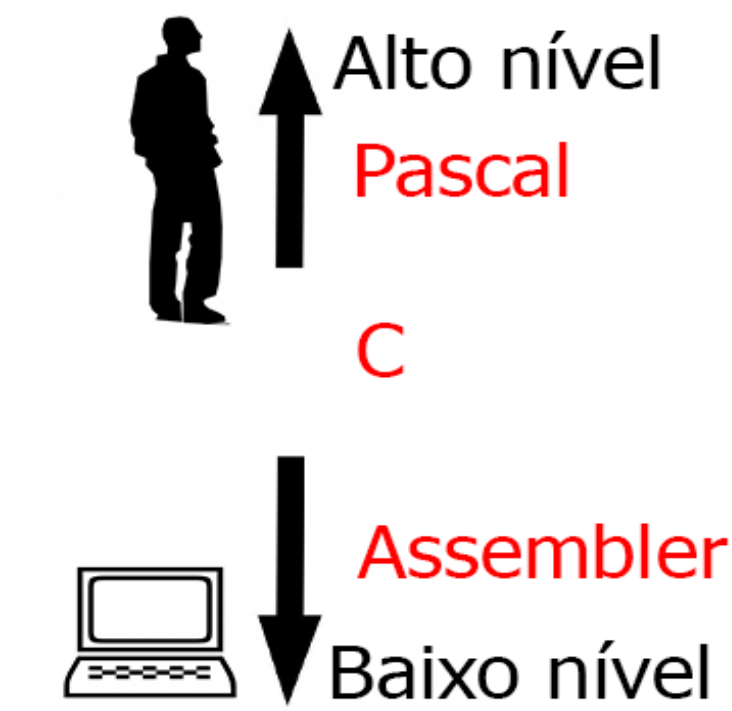
- C é uma linguagem compilada: lê todo o código fonte e gera o código objeto (ling. de máquina) uma única vez.

- Linguagens Interpretadas: lê o código fonte, traduz e executa cada vez que o programa for executado.

Influência da Linguagem C



Alto nível x Baixo Nível



Baixo nível

```
00000000                push    ebp
00000001                mov     ebp, esp
00000003                movzx   ecx, [ebp+arg_0]
00000007                pop     ebp
00000008                movzx   dx, cl
0000000C                lea     eax, [edx+edx]
0000000F                add     eax, edx
00000011                shl     eax, 2
00000014                add     eax, edx
00000016                shr     eax, 8
00000019                sub     cl, al
0000001B                shr     cl, 1
0000001D                add     al, cl
0000001F                shr     al, 5
00000022                movzx   eax, al
00000025                retn
```

Alto nível

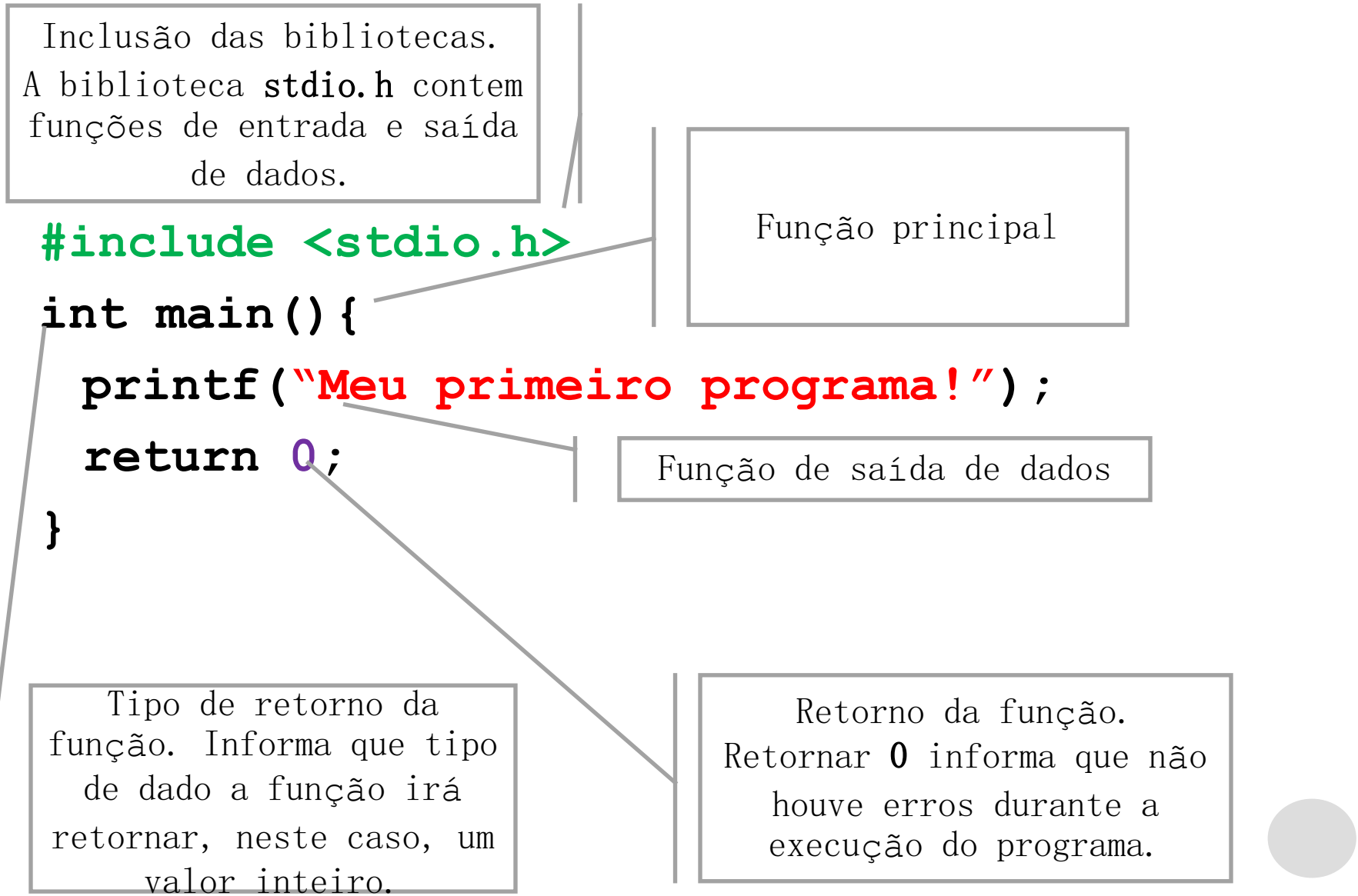
```
program numero;

var
x,y:integer;
begin
  read(x,y);
  if(x > y)
      write('X maior');
   if(x < y)
      write('X menor');
end.
```

Estrutura de um programa em C

```c
#include <stdio.h>
int main(){
  printf("Meu primeiro programa!");
  return 0;
}
```

Para salvar seu programa em C, bastar nomear o arquivo em **nome-programa.c**


Inclusão das bibliotecas.
A biblioteca stdio.h contem funções de entrada e saída de dados.
#include <stdio.h>
int main(){
printf("Meu primeiro programa!");
return 0;
}
Função principal
Função de saída de dados
Tipo de retorno da função. Informa que tipo de dado a função irá retornar, neste caso, um valor inteiro.
Retorno da função. Retornar 0 informa que não houve erros durante a execução do programa.

Função de Entrada de Dados

Leitura de dados via teclado.

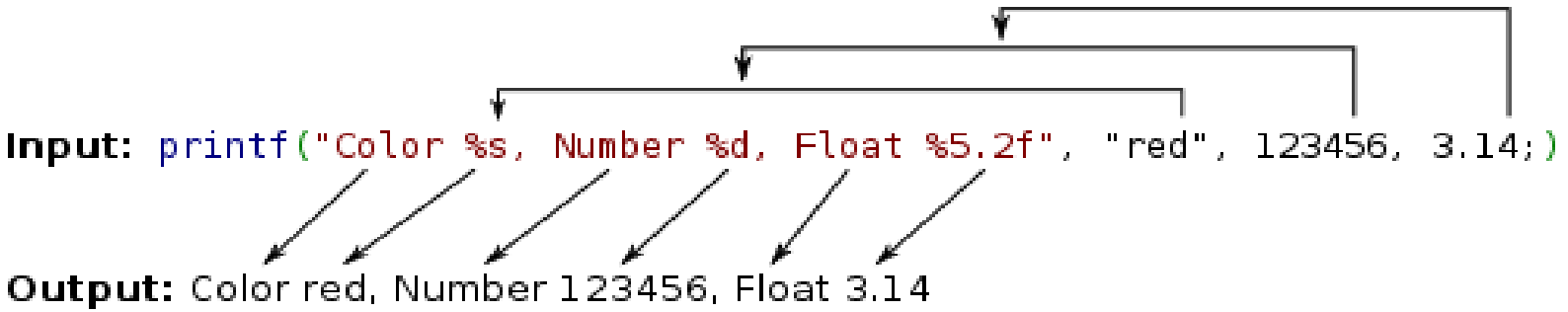
```
int scanf( char *, ...);
Ex: scanf("%d %f", &idade, &altura);
```

Para sequência de caracteres (%s), o caracter **&** não deverá ser usado.

Função de Saída de Dados

Apresenta dados no monitor

int printf(char *, ...);



String de Controle

String	Tipo de dados
%c	Caracter
%d	Inteiro
%e	Numero ou notação cientifica
%f	Ponto flutuante
%o	Octal
%x	Hexadecimal
%s	String (Cadeia de Caracteres)
%lf	Double

Caracteres Especiais

Caractere	Saída
\t	Tabulação
\n	Nova linha (Enter)
\a	Campainha
\’	Apostrofo
****	Barra invertida

Exemplo 01

```c
#include <stdio.h>
int main(){
  int x;
  x = 10;
  printf("Valor de X: %d\n", x);
  printf("Endereco de X: %p", &x);
  return 0;
}
```

Operador de Endereço &

- um **endereço de memória** é o nome que o computador usa para identificar uma variável.
- toda variável ocupa uma **área de memória e seu endereço** é o primeiro byte por ela ocupado
- Quando usamos **&** precedendo uma variável estamos falando do **endereço** desta variável na **memória**

EXEMPLO 02

```c
#include <stdio.h>
int main(){
  int x;
  x = 10;
  printf("Valor de X: %d", x);
  printf("Endereco de X: %p", &x);
  return 0;
}
```

SAIDA

```
Valor de X: 10
Endereco de X: 2686788
```

EXEMPLO 03

```
#include <stdio.h>
int main(){
  /*COMENTARIO: Declaração de variável do
  tipo inteiro*/
  int idade;
  printf(“Digite sua idade");
  scanf(“%d”, &idade);
  printf(“Idade %d“, idade);
  return 0;
}
```

EXEMPLO 04

```
#include <stdio.h>
int main(){
  int x, y;
  printf(“Digite valores para X e Y:");
  scanf(“%d %d”, &x, &y);
  printf(“A soma de %d + %d = %d“, x, y, x
  + y);
  return 0;
}
```

TIPOS DE DADOS

- Em C, como na maioria das linguagens, os dados são divididos tipos: inteiro, real, caracter, etc.
- Esta divisão se deve basicamente ao número de bytes reservados para cada dado. Cada tipo de dado possui um intervalo de valores permitidos.

Tipo	Tamanho	Intervalo	Uso
Char	1 byte	-128 a 127	Numero muito pequeno e caracter ASCII
Int	2 bytes	-32768 a 32767	Contator, controle de laço
Float	4 bytes	3.4e-38 a 3.4e38	Real (Precisão de 7 dígitos)
Double	8 bytes	1.7e-308 a 1.7e308	Científico (Precisão de 15 dígitos)

DECLARAÇÃO DE VARIÁVEIS

- Sintaxe para declaração de variáveis:

```
tipo_de_dado nome_variável [,
  nome_variavel_2];
```

- Exemplo:

```
int idade;
float altura, peso;
char sexo;
double bignum;
```

- quantos caracteres quiser (32);
- comece com letras ou sublinhado:
  - Seguidos de letras, números ou sublinhados
- C é *case sensitive*:
  - `peso <> Peso <> pEso`
- não podemos definir um identificador com o mesmo nome que uma palavra chave
  - `auto static extern int long if while do`

INICIALIZAÇÃO DE VARIÁVEIS

- É possível, em C, declarar uma variável e já armazenar nela um valor inicial. Chamamos este procedimento de inicialização de uma variável.

- Exemplo:

```
int i = 5, j = 4;
float pi = 3.14;
```

CONVERSÃO DE TIPO

- É uma técnica utilizada para modificar um determinado tipo de dado representado por uma variável.

  A sintaxe utilizada é a seguinte:

  (tipo) variável

EXEMPLO 05

```c
#include <stdio.h>
int main(){
  float media = 7.5;
  int x = (int) media;
  printf("Valor da media: %f", media);
  printf("Valor da X: %d", x);
  return 0;
}
```

Operadores Aritméticos

Operador	Ação
+	Adição
*	Multiplicação
/	Divisão
%	Resto da divisão inteira
-	Subtração (unário)
--	Decremento
++	Incremento

Operadores Relacionais e Lógicos

Operador	Ação
>	Maior que
>=	Maior ou igual que
<	Menor que
<=	Menor ou igual que
==	Igual a
!=	Diferente de
&&	Condição "E"
||	Condição "OU"
!	Não

Operadores incrementais e decrementais

São operadores que **adicionam** ou **subtraem** uma **unidade** do conteúdo de uma variável.

Ex:

```
int x = 10;
```

x++, equivale a **x = x + 1;**

++x, equivale a **x = x + 1;**

x--, equivale a **x = x - 1;**

--x, equivale a **x = x - 1;**

E QUAL A DIFERENÇA?

- No operador pós-incremental (**x++**), o valor da variável é retornado e após adicionado uma unidade a variável (+1).
- No operador pré-incremental (**--x**), é adicionado uma unidade (+1) ao valor da variável e após este valor é retornado.

Exemplo 06

```c
#include <stdio.h>
int main() {
 int x = 5;
 printf("%d", x);
 printf("%d", ++x);
 x--;
 printf("%d", x);
 return 0;
}
```

Falso = 0; Verdadeiro <> 0;

- Em C, todo resultado de uma comparação que for **falso** será **0**, e todo resultado **verdadeiro** será diferente de **0**.

FALSO	ZERO. (0)
VERDADEIRO	Qualquer valor diferente de ZERO. (1)

Exemplo 07

```c
#include <stdio.h>

int main() {
 printf("%d", 5 < 2);
 printf("%d", 10 == 10);
 printf("%d", !(11 != 7));
 return 0;
}
```

Erros no código?

```c
#include <stdio.h>;
int main(){
  int x, y;
  printf("Digite um valor para X: ");
  scanf("%d", x);
  printf("Valor digitado: %d", y);
}
```

Estude!

- Leia o primeiro capitulo do Livro Linguagem C descomplicada. (pág. 9 a 30)

- Próxima aula
  - Funções