



INSTITUTO FEDERAL
SANTA CATARINA

Programação de Computadores I - PRG1

Engenharia Elétrica

Edilson Hipolito da Silva

edilson.hipolito@ifsc.edu.br - <http://www.hipolito.info>

Aula 07 - Introdução ao C

Roteiro



INSTITUTO FEDERAL
SANTA CATARINA

- Introdução ao C
- Exercícios



Introdução a linguagem C

- Linguagem C:
 - Desenvolvida por Dennis Ritchie nos laboratórios da AT&T Bell (EUA) no início dos anos 70
 - C é uma das linguagens de maior aceitação:
 - Portabilidade (Compiladores disponíveis para PC's, Mainframes, etc.)
 - Reúne tanto características de alto nível quanto de baixo nível (muitas vezes chamada de nível médio)



Turbo C vs. Turbo Pascal

- Termos básicos similares
- Algumas funções semelhantes
- C tem maior controle direto sobre o computador
- A linguagem C é “case sensitive”



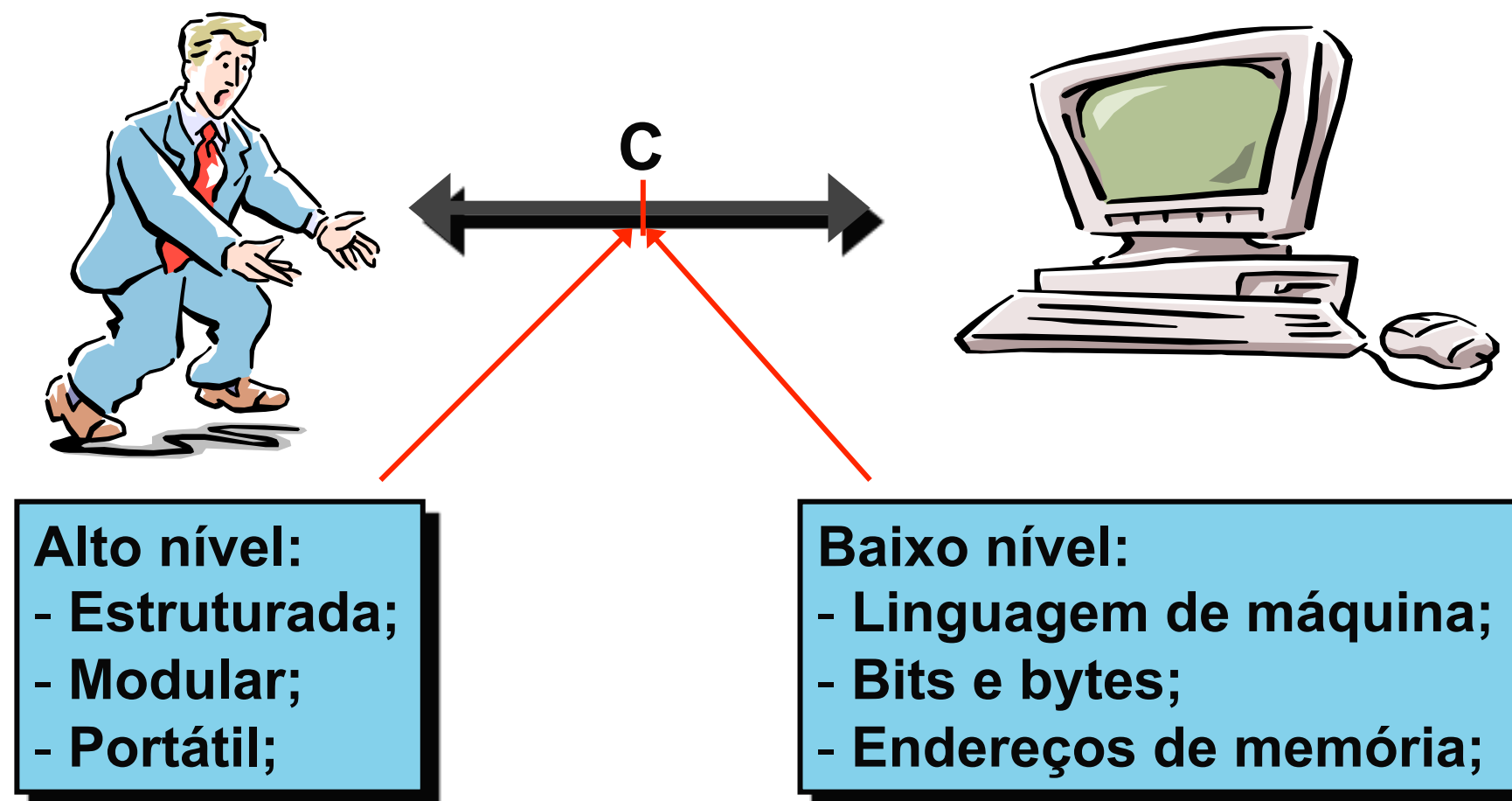
Introdução a linguagem C

- A linguagem C é utilizada na criação de diversos tipos de aplicação, tais como:
 - Sistemas Operacionais (UNIX)
 - Processadores de Texto
 - Planilhas Eletrônicas
 - Sistemas Gerenciadores de BD
 - Processadores Gráficos
 - Problemas de Engenharia
 - Compiladores



Introdução a linguagem C

- C: linguagem de nível médio





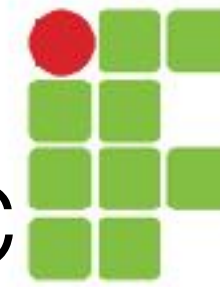
Introdução a Linguagem C

- Primeiros passos na Linguagem C:
 - C é case **sensitive**:
 - Pode-se declara quatro variáveis diferentes com os nomes Soma, soma, SOMA e SoMa, embora não seja recomendável na maioria das vezes;
- Todas as palavras reservadas da linguagem C devem ser descritas em letra minúscula, assim temos:
 - **for** : palavra reservada que rotula a estrutura de repetição “para”;
 - **if** : palavra reservada que rotula a estrutura de seleção “se”;
 - For : poderia ser um nome de variável;
 - If : poderia ser um nome de variável;



Introdução a Linguagem C

- Estrutura básica de um programa em C:
- Todo programa em C possui no mínimo **uma função**
- A função **main()** é obrigatória e é a primeira a ser executada. É a função principal, o ponto de partida do programa que será criado
- Por isso, a noção de função e de programa freqüentemente se confundem quando se programa em linguagem C



Forma geral de um programa em C

inclusão de arquivos de cabeçalho

declaração de variáveis globais

main () declaração da função principal (obrigatória). Os parênteses indicam que trata-se de uma função

{ toda função começa com o símbolo “{”, equivale ao “início” (begin) de um bloco de instruções

declaração de variáveis locais

...

comandos;

...

} toda função termina com o símbolo “}”, equivale ao “fim” (end) de um bloco de instruções

- As instruções descritas no programa terminam sempre com “;”, como na linguagem Pascal.

O menor programa possível em C



INSTITUTO FEDERAL
SANTA CATARINA

```
main () {
```

```
}
```

- É um programa vazio, que não contém instruções para executar;
- Formatação dos programas: é livre, mas convém **manter a legibilidade**;

```
main ()  
{  
}
```

Bons exemplos

```
main ()  
{ }
```

```
main  
(  
)  
{  
}
```

```
main  
(  
{  
}
```

Maus exemplos



Introdução a Linguagem C

- Comentários: podem ser incluídos em qualquer parte do programa.
 - Se ocupar apenas uma linha, basta que a mesma seja iniciada com “//”;
 - Se ocupar mais de uma linha, o comentário deve iniciar com o símbolo “/*” e terminar com o símbolo “*/”;
- Palavras reservadas da Linguagem C:

auto	break	case	char	const	continue	default
do	double	else	enum	extern	float	for
goto	if	int	long	main	register	return
short	signed	sizeof	static	struct	switch	typedef
union	unsigned	void	volatile	while		



Programação Estruturada

- **Um programa é composto por blocos elementares de código que se interligam através de três mecanismos básicos:**
 - **Sequência:** implementa os passos para descrever qualquer programa.
 - **Seleção/Decisão:** possibilidade de selecionar/alterar o fluxo de execução do código baseado em ocorrências lógicas.
 - **Iteração/Repetição (Laços):** permite a execução repetitiva de blocos do programa.
- Esses mecanismos se traduzem em estruturas nas linguagens de programação.
- Cada uma destas estruturas tem um ponto de início e um ponto de término de execução.

Exemplo



```
media.c
1 #include <stdio.h>
2
3 int main(){
4     float n1, n2, m;
5
6     printf("Digite as duas notas:\n");
7     scanf("%f", &n1);
8     scanf("%f", &n2);
9
10    m = (n1 + n2) / 2;
11
12    printf("Média: %.2f\n", m);
13    if(m >= 7.0){
14        printf("Aprovado\n");
15    }
16    else{
17        printf("Reprovado\n");
18    }
19 }
```

Estrutura de um programa em C



INSTITUTO FEDERAL
SANTA CATARINA

```
primeiro.c ✕  
1 #include <stdio.h>  
2 int main ()  
3 {  
4     printf ("Olá Mundo!");  
5     return 0;  
6 }
```

Estrutura de um programa em C



INSTITUTO FEDERAL
SANTA CATARINA

Avisa ao compilador
que as funções de
entrada e saída de
dados da biblioteca
stdio

A função main é
o corpo principal
do programa.
int main() { ... }
Todo programa
deve ter um main.
As chaves { e }
marcam o **início** e
o **fim** da função.

```
primeiro.c x
1 #include <stdio.h>
2 int main ()
3 {
4     printf ("Olá Mundo!");
5     return 0;
6 }
```

O programa usa a função
printf(...) para imprimir
a mensagem no console.
Texto deve ser entre aspas:
"Olá Mundo!"

Avisa o SO que o
programa terminou
sem problemas.



Inicialmente temos

- $f(x) = x^2 + 2x + 10$



Um código intermediário

```
f(x){
```

```
    (x * x) + (2 * x) + 10;
```

```
}
```

```
main() {
```

```
    "resultado: ", f(5);
```

```
}
```

Código em C



INSTITUTO FEDERAL
SANTA CATARINA

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

```
float f(float x){
```

```
    return (x * x) + (2 * x) + 10;
```

```
}
```

```
int main() {
```

```
    printf("resultado: %f", f(5));
```

```
    return 0;
```

```
}
```



Variáveis e Constantes

- **Variáveis e constantes** são os **elementos básicos**.
- Uma variável corresponde a um **espaço reservado** na memória para **armazenar** um determinado tipo de dado.
- Variáveis devem receber **nomes significativos** que possam ser **referenciados** e modificados.
- É preciso **declarar** uma variável **antes de usá-la**.
- Declarações **especificam** de que **tipo** são as variáveis e as vezes um **valor inicial**.
- **Tipos podem ser por exemplo**: inteiros, reais, caracteres, etc.
- **Expressões** combinam **variáveis** e **constantes** para calcular novos valores.



Nomes das variáveis

- **Algumas regras básicas para a nomeação de variáveis:**
 - Todo nome só pode conter letras e dígitos;
 - O caractere "_" é contado como uma letra;
 - Todo primeiro caractere deve ser sempre uma letra;
 - Letras maiúsculas e minúsculas são consideradas caracteres diferentes → case sensitive.
 - Palavras reservadas não podem ser usadas como nome de variáveis.
- **É uma boa prática escolher nomes que significam alguma coisa e indiquem a função da variável.**
 - **Por exemplo:** valor, soma, total, nome, raio.



Declarando variáveis

- Para serem usadas, as variáveis precisam ser declaradas de modo que o compilador possa reservar espaço na memória para o valor a ser armazenado.
- A forma geral de uma declaração é:
 - tipo lista_de_variaveis;
- **Exemplos:**

```
int i;
```

```
int a, b, c;
```

```
unsigned int dia, mes, ano;
```

```
double salario;
```



Tipos de dados

- **Os dados em C podem assumir cinco tipos básicos:**
- **char:** Armazena um caractere. Caracteres geralmente são armazenados em códigos (código ASCII) e ocupam um byte.
- **int:** O valor armazenado é um número inteiro. Números inteiros são armazenados em 32 bits.
- **float:** Número em ponto flutuante de precisão simples, normalmente utilizam 32 bits. São os conhecidos números reais.
- **double:** Número em ponto flutuante de precisão dupla, armazenado em 64 bits.
- **void:** Este tipo serve para indicar que um resultado não tem um tipo definido. Tipo vazio.



Modificadores de tipos

- **Modificadores são palavras que alteram o tamanho do conjunto de valores que o tipo pode representar.**
- **Por exemplo:** quero armazenar um inteiro que sei que não tem sinal.
- Modificadores:
 - **unsigned:** Pode ser aplicado aos tipos **int** e **char** e faz com o bit de sinal não seja usado, ou seja o tipo passa a ter um bit a mais, ampliando a sua capacidade.
 - **long:** Pode ser aplicado aos tipos **int** e **double** aumentando o número de bytes reservado para armazenamento de dados.

Tabela de combinações permitidas



INSTITUTO FEDERAL
SANTA CATARINA

Tipo	Bytes	Faixa Mínima
char	1	-127 a 127
unsigned char	1	0 a 255
signed char	1	-127 a 127
int	4	-2.147.483.648 a 2.147.483.647
unsigned int	4	0 a 4.294.967.295
signed int	4	-2.147.483.648 a 2.147.483.647
short int, short	2	-32.768 a 32.767
unsigned short int	2	0 a 65.535
signed short int	2	-32.768 a 32.767
long int, long	4	-2.147.483.648 a 2.147.483.647
signed long int	4	-2.147.483.648 a 2.147.483.647
unsigned long int	4	0 a 4.294.967.295
long long int long long	8	-9.223.372.036.854.775.808 a 9.223.372.036.854.775.807
signed long long int signed long long	8	-9.223.372.036.854.775.808 a 9.223.372.036.854.775.807
unsigned long long int unsigned long long	8	0 a 18.446.744.073.709.551.615
float	4	oito dígitos de precisão
double	8	16 dígitos de precisão
long double	12	16 dígitos de precisão



Atribuição de valores

- Após ser declarada, uma variável pode receber valores.
- O operador de atribuição é: =
- Indicando que o resultado da expressão à direita do operador será atribuído à variável.

- **Atribuições durante a declaração:**

`int i = 0, j = 10;`

`char c = 'd';`

`float raio = 2.54;`

`double precisao = 0.00001 L;`

- **Atribuições pós declaração:**

`int i;`

`float raio ;`

`j = 10;`

`raio = 2.54;`



Constantes

- Constantes não podem ser alteradas como as variáveis.

```
1 #include <stdio.h>
2
3 #define PI 3.14159265
4 #define NEW_LINE '\n'
5
6 int main ()
7 {
8     printf("Valor de PI: %f \n", PI);
9     printf("Valor de PI: %.|f \n", PI);
10    printf("Valor de PI: %.8f \n", PI);
11
12    printf ("Olá Mundo!%c", NEW_LINE);
13    return 0;
14 }
```



Funções de entrada e saída

- **A função `printf(...)` escreve dados na saída padrão, normalmente o terminal.**
- O protótipo da função é:
- **`int printf(controle, arg1, arg2, ...);`**
- Onde os argumentos `arg1`, `arg2`, ... são impressos de acordo com o formato indicado pela cadeia de caracteres que compõe `controle`.
- O formato permite que os resultados possam ser apresentados de diversas maneiras
- A função retorna o número de caracteres impressos.
- No caso de um erro um valor negativo é retornado.

Caracteres para saída formatada



INSTITUTO FEDERAL
SANTA CATARINA

Código	Comentário
%c	Caracter simples
%d	Inteiro decimal com sinal
%i	Inteiro decimal com sinal
%E	Real em notação científica com E
%e	Real em notação científica com e
%f	Real em ponto flutuante
%G	%E ou %f, o que for mais curto
%g	%g ou %f, o que for mais curto
%o	Inteiro em base octal
%s	Cadeia Caracteres
%u	Inteiro decimal sem sinal
%x	Inteiro em base hexadecimal (letras minúsculas)
%X	Inteiro em base hexadecimal (letras maiúsculas)
%p	Endereço de memória
%%	Imprime o caractere %



Exemplo printf()

```
int x = 10;
```

```
float y = 5.0;
```

```
printf(“%d%f\n”,x,y);
```



Funções de entrada e saída

- **A função `scanf(...)` pode ser utilizada para entrada de dados a partir do teclado.**
- **Protótipo:**
 - `scanf(controle, &arg1, &arg2, ...);`
- Na `scanf` os argumentos que vem depois do controle são endereços das variáveis que irão receber os valores lidos e não como no `printf` que são as próprias variáveis. O `&` indica que estamos referenciando o endereço. Por exemplo:
 - `scanf("%d %d", &a, &b);`
- **Lê dois valores inteiros que serão armazenados na variável `a` e `b`, nos endereços indicados por `&a` e `&b` respectivamente.**



Funções de entrada e saída

- Exemplo incluindo outros tipos de variáveis:

```
1#include <stdio.h>
2int main (void){
3
4    int i;
5    float f;
6    char c;
7
8    printf("Digite os valores: \n");
9    scanf("%d %f %c", &i, &f, &c);
10
11    printf("Valores digitados: \n");
12    printf("Inteiro: %d\n", i);
13    printf("Float: %f\n", f);
14    printf("Caracter: %c\n", c);
15
16    return 0;
17}
```

Exemplo



INSTITUTO FEDERAL
SANTA CATARINA

```
#define PI 3.1415

int main(void){
    int area, raio;

    printf( "Digite o raio:\n" );
    scanf( "%f" , &raio);

    printf("Area: %f \n", area);
}
```



Exemplo



INSTITUTO FEDERAL
SANTA CATARINA

```
#define PI 3.1415

int main(void) {
    int area, raio;

    printf("Digite o raio:\n");
    scanf("%f", &raio);

    area = 3.1415 * raio * raio;

    printf("Area: %f \n", area);
}
```

Exercício



INSTITUTO FEDERAL
SANTA CATARINA

-
- Indique os nomes de variáveis que são válidos. Justifique os nomes inválidos.
 - tempo
 - nota_final
 - us\$
 - char
 - 2dias
 - teste 1
 - raio.do.circulo

Exercício



INSTITUTO FEDERAL
SANTA CATARINA

- Leia um número e imprima seu sucessor e seu antecessor.
- Faça um algoritmo que receba 3 números, calcule e mostre a multiplicação desses números.
- Faça um programa que receba duas notas, calcule e mostre a média ponderada dessas notas, considerando peso 2 para a primeira nota e peso 3 para a segunda nota.

Dúvidas?



INSTITUTO FEDERAL
SANTA CATARINA

