



# SCC0221 – Introdução à Ciência de Computação I

---

**Prof.: Dr. Rudinei Goularte**

(rudinei@icmc.usp.br)

## Visão Geral da Linguagem C

Instituto de Ciências Matemáticas e de Computação - ICMC  
Sala 4-229



# Características de C

---

- Características Gerais
  - linguagem de nível *médio*.
  - não é uma linguagem fortemente tipada.
  - uso intensivo de ponteiros.
  - definição de blocos { }.
  - pré-processador.
  - não define operações de entrada e saída.
  - funções retornam valor e podem ser chamadas recursivamente.



# Características de C

---

- Vantagens:

- C é uma linguagem pequena.
- C é a linguagem nativa dos sistemas baseado em UNIX.
- C é portátil\*.
- Vasto conjunto de operadores, que podem acessar a máquina no nível de bits.
- C é modular.
- C é base para C++ e Java.



# Características de C

---

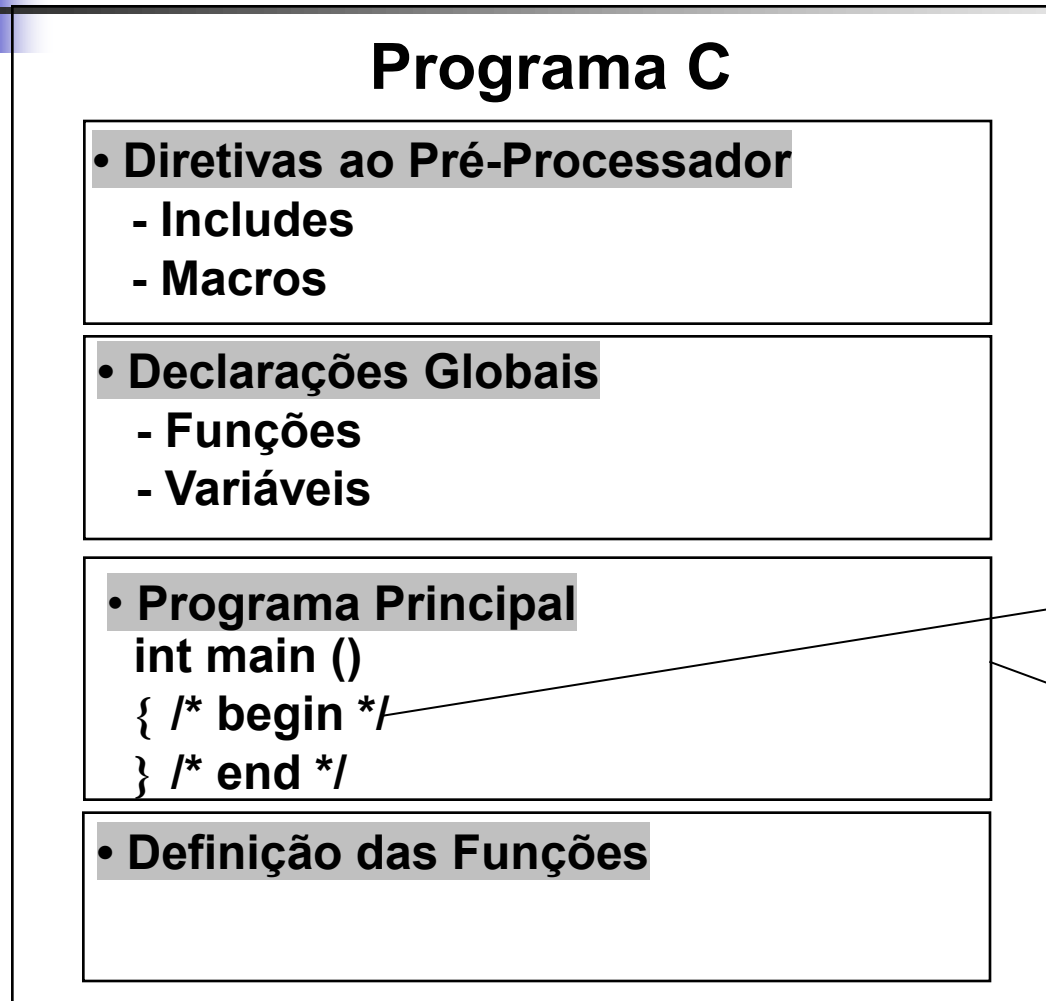
- Desvantagens:
  - Sintaxe complicada.
  - Não possui verificação automática de limites de arrays.
  - Faz múltiplos usos de alguns símbolos, como \* e =.



# Visão Geral da Linguagem C

---

# Estrutura de um programa em C



/\* Isto é um  
comentário \*/

Programa  
mínimo em C



# Primeiro Programa

---

## ■ Exemplo clássico:

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

```
int main(void)
```

```
{
```

```
    printf("Hello World!\n");
```

```
    return(0);
```

```
}
```



# Primeiro Programa

---

```
#include <stdio.h>
int main(void)
{
    printf("hello, world!\n");
    return(0);
}
```

- *todo* programa C tem que ter uma função chamada **main( )**. É aqui que se inicia a execução do programa.
- em um programa pequeno, todo o algoritmo pode ser escrito dentro de *main()*.
- programas estruturados consistem em uma hierarquia de funções dentre as quais *main()* é aquela de mais alto nível.





# Compilando Programas em C

---



# Sistemas de Programação

---

## ■ COMPILADOR

- Traduz os comandos de uma linguagem de programação para linguagem de máquina.
- Recebe como entrada um código-fonte. Devolve como saída um arquivo contendo o código-objeto.
- Código-objeto é a tradução do código-fonte para a linguagem de máquina.

## ■ INTERPRETADOR

- Lê, interpreta e executa, uma instrução do código-fonte do programa por vez.
- Nenhuma fase intermediária de compilação é necessária.
- A execução do programa interpretado requer que o interpretador da linguagem esteja sendo executado no computador, ao mesmo tempo em que o programa em si.



# Sistemas de Programação

---

- MONTADOR

- Traduz os comandos simbólicos de uma linguagem de montagem para linguagem de máquina.
- Linguagem de montagem é uma representação simbólica da linguagem de máquina. Ex.: Assembly.

- LIGADOR (Linker)

- Liga todos os códigos-objeto de um programa em um único código executável.

## Arquivo-FONTE

```
/* **** */
/* Primeiro exemplo  arq exemplo1.c */
/* **** */
#include <stdio.h>

/* C padrão de Entrada/Saída */
/* **** */
main () /* Comentários em C */
{
    printf ("exemplo nro %d em C !", 1);
    printf ("\n depois o %d ! \n", 2);
    printf ("criatividade em baixa \n");
}
```

*source-file*

Compilador

## Arquivo-OBJETO

```
1111000101010010
0101100010010100
1110001100010000
1110000000010000
```

*object-file*

Outros Arquivos  
OBJETO/Bibliotecas

```
0101001010000000
1111000101010010
0101100010010100
1110001100010000
1100010100000000
```

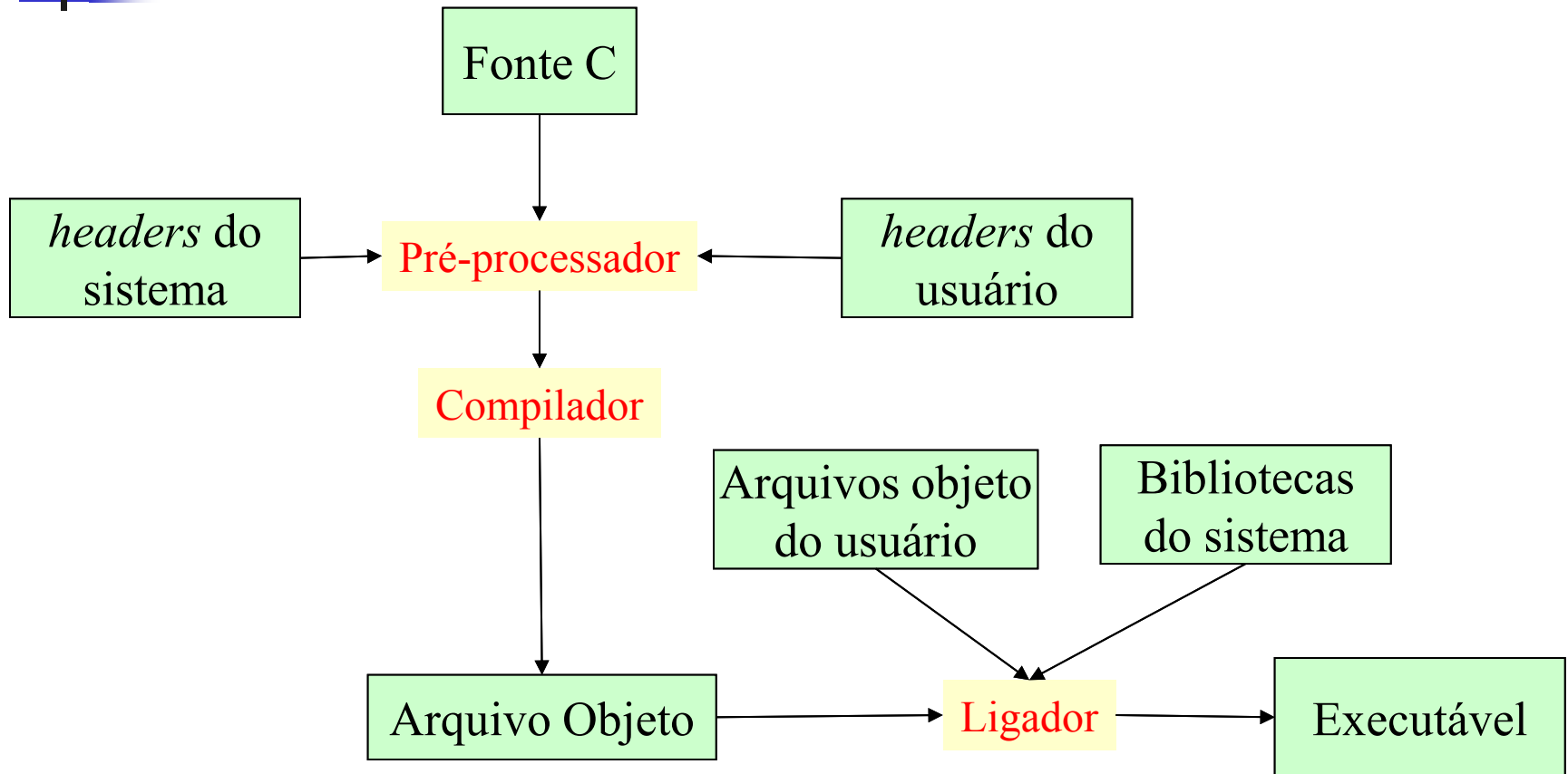
*libraries*

Link-editor

```
1111000101010010
0101100010010100
1110001100010000
1110000000010000
0000000010001010
1100010100000000
0011000100000010
1110000100000011
```

## Arquivo-EXECUTÁVEL

# Fluxo do Compilador C





# Quê softwares usar?

---

- Onde “escrever” programas em C?
  - Qualquer editor de texto.
  - Ambientes de programação para C.
- Qual compilador/ambiente usar?
  - Existem diversos disponíveis.
  - Uma boa dica é usar um compilador que dê suporte à compilação seguindo o padrão ANSI C.
  - Exemplos: DevC e CodeBlocks, para Windows; GCC, para Linux.



# Palavras Reservadas

---

Padrão ANSI (American National Standards Institute):

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



# Variáveis e Constantes

---





# Constantes

---

- Constantes

- São valores fixos que não podem ser modificados pelo programa.

- Exemplos:

- Caracteres: `'a'`, `'\n'`, `'9'`
- Valores inteiros: `123`, `1`, `1000`, `-23`
- Inteiros longos: `35000L`, `-45L`
- Inteiros sem sinal: `1000U`, `234U`, `4365U`
- Reais: `123.45F`, `3.1415e-10F`
- Reais de precisão dupla: `123.45`, `-0.91254`
- Strings: `"abcd"`, `"isto é uma string!"`, `"Av. São Carlos, 2350"`
- Octais: `012`: equivale a 10 em decimal
- Hexadecimais: `0xA`: equivale a 10 em decimal



# Constantes

## Barra invertida

- \a: bip
- \b: backspace
- \n: newline
- \t: tab horizontal
- \': apóstrofe
- \": aspa
- \\: backslash
- \f: form feed
- \r: return
- \0: nulo

- Constantes caracter entre aspas simples funciona para a maioria dos casos.
- Mas, existem alguns caracteres especiais que necessitam da barra invertida. Ao lado temos uma lista dos mais comuns.



# Variáveis

---

- Variáveis.

- Em um programa C estão associadas a posições de memória que armazenam informações.
- Toda variável deve estar associada a um identificador.
- Palavras-chave de C não podem ser utilizadas como nome de variáveis: int, for, while, etc...
- C é case-sensitive:
  - contador ≠ Contador ≠ CONTADOR ≠ cOntaDor



# Exercício

---

- Escolha a opção que inclui somente nomes válidos para variáveis na linguagem C.
  - a) i, j, int, obs
  - b) 9xy, a36, x\*y, --j
  - c) 2\_ou\_1, \fim, \*h, j
  - d) If, a\_b\_2, H789, \_yes
  - e) Nenhuma das opções anteriores



# Tipos de Datos Básicos

---



# Tipo de Dado

---

- É um método para interpretar o conteúdo da memória do computador. Define a quantidade de memória que deve ser reservada para uma variável e como os bits devem ser interpretados.
  - O *tipo* de uma variável define os valores que ela pode assumir e as operações que podem ser realizadas com ela.
- Ex:
  - variáveis tipo *int* recebem apenas valores inteiros.
  - variáveis tipo *float* armazenam apenas valores reais.



# Tipos Básicos em C

---

- Os tipos de dados básicos, em C, são 5:
  - Caracter: **char**
    - Exemplos: `'a'`, `'1'`, `'+'`, `'$'`, ...
  - Inteiro: **int**
    - Exemplos: `-1`, `1`, `0`, ...
  - Real: **float**
    - Exemplos: `25.9`, `-2.8`, ...
  - Real de precisão dupla: **double**
    - Exemplos: `25.9`, `-2.8`, ...
  - Sem valor: **void**
- Todos os outros tipos são derivados desses 5.



# Tipos Básicos em C

---

- Existem os tipos criados pelo usuário, a partir dos tipos básicos.
  - Tais tipos são primeiro definidos.
  - Uma definição de tipo cria um modelo que pode ser usado para se declarar variáveis.





# Abrangência dos Dados

---

- Depende da Arquitetura do Processador
- Padrão ANSI define apenas a faixa mínima de valores de cada tipo, mas não seu tamanho

Tipo	Tamanho(bytes)*	Abrangência		
char	1	-128	a	127
unsigned char	1	0	a	255
signed char	1	-128	a	127
int	2	-32768	a	32767
unsigned int	2	0	a	65535
signed int	2	mesmo que int		
short int	2	mesmo que int		
unsigned short int	2	0	a	65535
signed short int	2	o mesmo que short int		
long int	4	-2.147.483.647	a	2.147.483.647
signed long int	4	o mesmo que long int		
unsigned long	4	0	a	4.294.967.295
float	4	6 dígitos de precisão (-3,4·10 <sup>38</sup> a 3,4·10 <sup>38</sup> )		
double	8	10 dígitos de precisão (-1,7·10 <sup>308</sup> a 1,7·10 <sup>308</sup> )		
long double	10	10 dígitos de precisão (-3,4·10 <sup>4932</sup> a 3,4·10 <sup>4932</sup> )		

# Caracteres e o Tipo de Dado char



---

- Em C, quaisquer variáveis de tipo inteiro podem ser usadas para representar um caracter.
  - Em geral usa-se char e int para isso.
- Constantes como `'a'` e `'+'`, que nós “pensamos” como caracteres, são do tipo inteiro.



# Caracteres e o Tipo de Dado char

---

- Para representar caracteres, variáveis do tipo char podem ser usadas para armazenar pequenos valores inteiros.
- Cada char é armazenado na memória em um byte.
- $2^8 = 256$  valores possíveis.
- Código ASCII.



# Declaração de Variáveis

---



# Declaração de Variáveis

---

- A declaração de uma variável segue o modelo:

TIPO DE DADO identificador, ..., identificador;

- Ex:

```
int x, y, z;
```

```
float f;
```

```
unsigned int u;
```

```
long double df;
```



# Atribuição e Inicialização

---

- Operador de atribuição: =
  - Exemplo: `int a; a = 10;`
- Variáveis podem ser inicializadas na declaração:
  - `int a = 10;`
- Inicializar uma variável significar atribuir à mesma um valor inicial válido.
  - Ao se declarar a variável, a posição de memória da mesma contém um valor aleatório.



# Onde declarar variáveis?

---

- Em três lugares, basicamente:
  - Dentro de funções: variáveis locais.
  - Na definição de parâmetros de funções: parâmetros formais.
  - Fora de todas as funções: variáveis globais.
- Exemplo no quadro.
  - Declaração de variáveis globais e locais a main.



# Escopo de Variáveis

---

- Escopo define **onde** e **quando** uma variável pode ser usada em um programa.
- variável declarada **fora** das funções – global - tem escopo em todo o programa:

```
#include <stdio.h>
int i = 0;           /* variavel global */
                    /* visivel em todo arquivo */
void incr_i() { i++;}
...
void main() { incr_i(); printf("%d", i);}
```





## 4.4 Escopo de Variáveis

---

- Escopo de bloco: é visível apenas no interior do bloco

```
...  
  
if (teste == TRUE) {  
    int i;  
    i = i+1;  
    ...  
}  
else { i = i - 1; /* erro: i não definida */  
    ...  
}  
...
```



## 4.4 Escopo de Variáveis

---

- Escopo de função: variável declarada na lista de parâmetros da função ou definida dentro da função.

- Ex: ...

```
void f (void) {  
    printf ("%d %d", i, j); /* erro: i e j não definidos */  
}  
int main (void) {  
    int i, j; /* i e j visíveis apenas dentro da função main */  
    f();  
    ...  
}
```



## Exercício 3

---

- O código abaixo está correto? Explique.

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



# Exercício 4

---

- O quê será impresso?

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

```
int i = 5;
```

```
int main (void){
```

```
    int i = 10;
```

```
    printf ("%d", i);
```

```
    return(0);
```

```
}
```



# Atribuições

---



# Atribuição de Variáveis

---

- Forma geral: `nome_da_variável = expressão;`
  - Atribui o valor de *expressão* (à direita de =) à variável à esquerda de =.
  - Expressão pode ser desde uma constante até uma expressão complexa.
- Múltiplas atribuições
  - C permite a atribuição de mais de uma variável em um mesmo comando:

$$x = y = z = 0;$$

- 
- As atribuições do programa abaixo estão corretas?

```
int main (void){  
    int i = 1; char c = 'A'; float f = 5.0;  
    c = i;  
    i = f;  
    f = c;  
    f = i;  
    return(0);  
}
```

# Conversões de Tipos na Atribuição

- Quando uma variável de um tipo é atribuída a uma de outro tipo, o compilador automaticamente converte o tipo da variável a direita de “=” para o tipo da variável a esquerda de “=”.
- Ex: (assumindo char de 1 byte)  
int i; char c; float f;  
  
c = i; /\* c recebe 8 bits menos significativos de i \*/  
i = f; /\* i recebe parte inteira de f \*/  
f = c; /\* f recebe os 8 bits de c convertidos para real \*/  
f = i; \* idem para inteiro i \*/





# Entrada e Saída Formatadas

---



# O Comando printf()

---

- Características:
  - Definido em `stdio.h`
  - Permite escrever dados em vários formatos.
  - Forma geral:
    - `printf ("string_de_controle", lista de variáveis);`
    - A string de controle pode conter:
      - Caracteres que serão impressos na tela.
      - Comandos de formato. Começam com o símbolo %
    - A lista de variáveis contém os nomes das variáveis cujos valores serão impressos de acordo com o formato especificado na string de controle.



# O Comando printf()

Comando	Formato
%c	Caractere
%d	Inteiro
%e	Notação científica
%f	Float
%o	Octal
%s	String
%u	Inteiro sem sinal
%x	Hexadecimal
%%	Escreve o símbolo %



# O Comando printf()

---

- Exemplo:  

```
int num = 5;  
printf("O número é %d", num);
```
- Outro exemplo:  

```
int num = 12;  
char ch = 'A';  
printf("Os dados são:");  
printf("\n");  
printf("o número %d ", num);  
printf("e o caractere %c", ch);
```
- Exercício 3: Obtenha a mesma mensagem usando só um printf.



# O Comando printf()

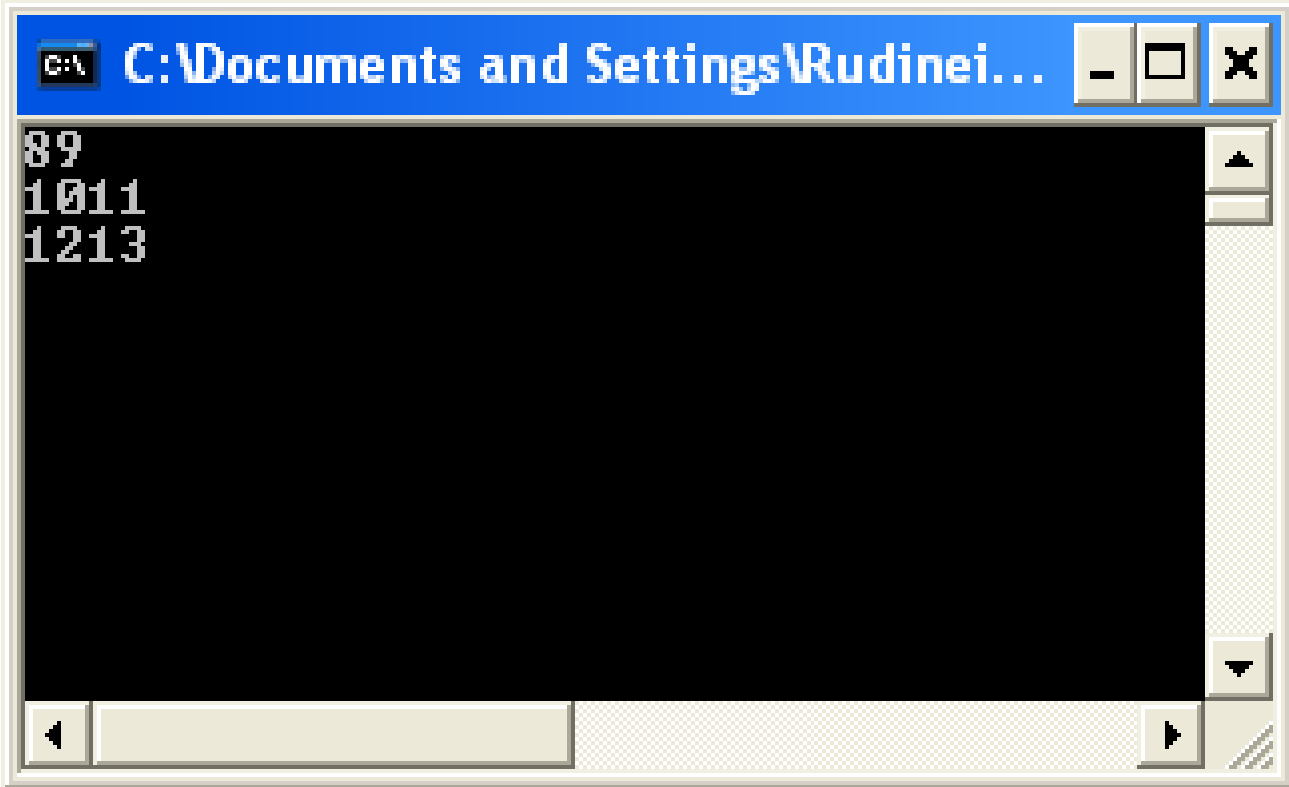
---

- Largura Mínima de campo
  - Número entre % e o código de formato.
  - Alinha à direita.
  - Exemplo:

```
int num = 8;  
printf ("%d",num); num = num +1;  
printf ("%d \n",num); num = num +1;  
printf ("%d",num); num = num +1;  
printf ("%d\n",num); num = num +1;  
printf ("%d",num); num = num +1;  
printf ("%d",num);
```



# O Comando printf()



```
C:\Documents and Settings\Rudinei...  
89  
1011  
1213
```



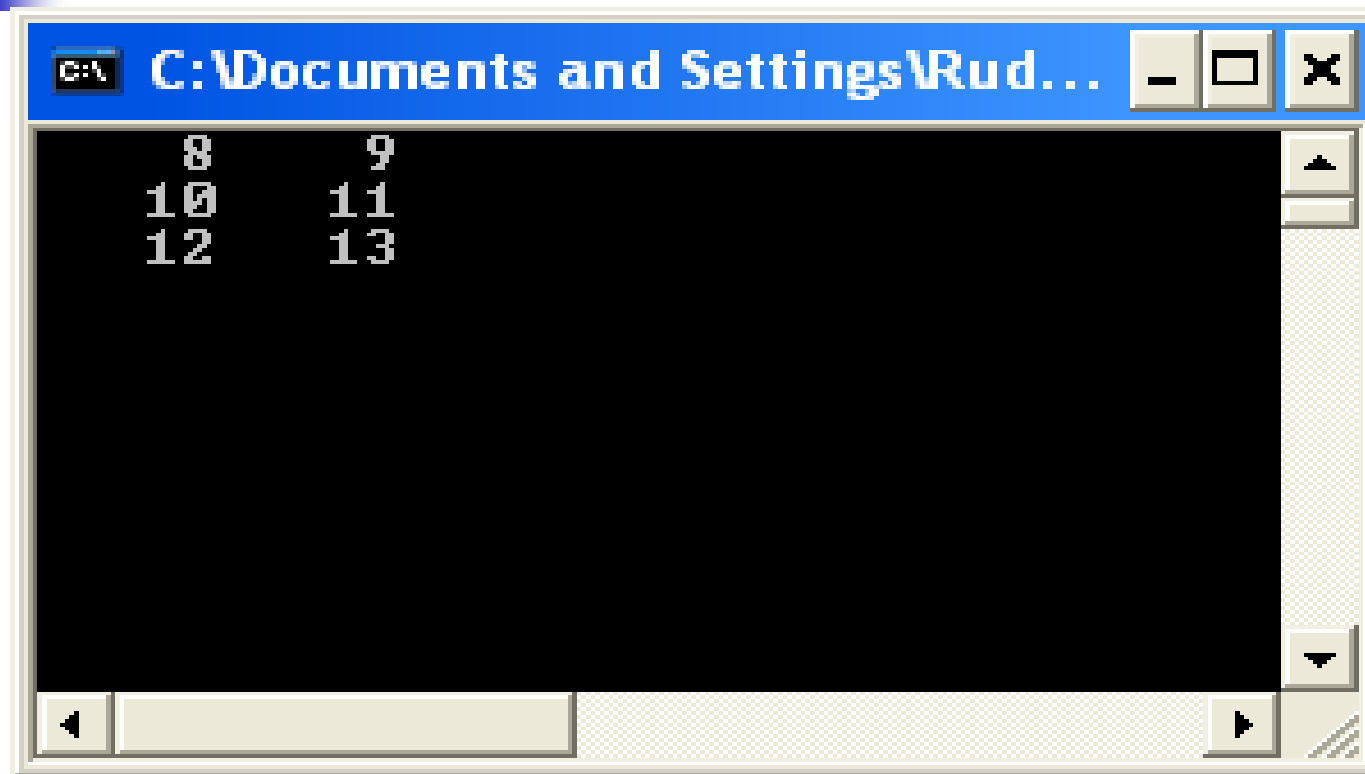
# O Comando printf()

---

- Largura Mínima de campo
  - Número entre % e o código de formato.
  - Alinha à direita.
  - Exemplo:

```
int num = 8;  
printf ("%5d",num); num = num +1;  
printf ("%5d \n",num); num = num +1;  
printf ("%5d",num); num = num +1;  
printf ("%5d\n",num); num = num +1;  
printf ("%5d",num); num = num +1;  
printf ("%5d",num);
```

# O Comando printf()



A screenshot of a Windows command prompt window. The title bar reads "C:\Documents and Settings\Rud...". The command prompt shows the output of three printf statements. The first line shows "8" and "9" aligned to the right. The second line shows "10" and "11" aligned to the right. The third line shows "12" and "13" aligned to the right. The numbers are displayed in a monospaced font.

```
C:\Documents and Settings\Rud...  
8      9  
10     11  
12     13
```

- Para alinhar à esquerda basta adicionar o sinal –
  - `printf ("% -5d", num);`





# O Comando printf()

---

- Especificador de Precisão
  - Um ponto seguido de um número inteiro.
  - Limita o número de casas decimais a serem impressas.
  - Exemplo:
    - `double num = 3.456789;`  
`printf("%.2f", num);`
    - Resultado: 3.45



# O Comando scanf()

---

- Características:
  - Definido em `stdio.h`
  - Permite ler dados, em vários formatos, vindos do teclado.
  - Forma geral:
    - `scanf ("string_de_controle", lista de variáveis);`
    - A string de controle pode conter:
      - Especificadores de formato.
  - A lista de variáveis contém os nomes das variáveis cujos valores serão lidos do teclado, no formato especificado, e armazenados, respectivamente, nas variáveis.



# O Comando scanf()

---

- Exemplo:
  - ```
int num;  
scanf("%d", &num);  
printf("%d", num);
```
- **IMPORTANTE:** scanf necessita que toda variável, exceto strings, usem o operador &.
- Outro exemplo:
  - ```
int num;  
char ch;  
scanf("%d %c", &num, &ch);  
printf("o número é %d \n", num);  
printf("e o caracter é %c", ch);
```



# O Comando scanf()

Comando	Formato
%c	Lê um caracter
%d	Lê um inteiro
%e	Número em ponto flutuante
%f	idem
%o	Octal
%s	String
%x	Hexadecimal
%u	Lê um inteiro sem sinal



# O Comando scanf()

---

- Lendo strings
  - O comando:  
`scanf ("%s", str);`
  - Lê uma string até encontrar um espaço em branco.
  - Coloca `'\0'` no fim da string.
  - Da entrada: *a/ô vocês!*, seria armazenado apenas *a/ô* em str.



## Para Saber Mais...

---

- Kelley, A.; Pohl, I. "A Book on C".  
Capítulos 0 e 1.