

Linguagem de Programação I



**Prof.: Rubens L. Cirino, DSc, PMP ,PRINCE2
HCMP 3G Expert Professional**

rlcirino@ime.uerj.br

Agenda

- Regras e Recomendações
- Introdução à Linguagem C
- Saída de Dados
- Tipos de dados em C
- Operadores e Expressões
- Entrada de Dados
- Desvio Condicional
- Laços
- Funções
- Vetores
- Ponteiros
- Estruturas de Dados
- Operação com arquivos em disco

Regras e Recomendações

- Avaliação

Uma avaliação (prova) e dois trabalhos

- Média = $(7 * \text{Nota da prova} + 3 * (\text{Trabalhos e Lista})) / 10$
- Critério de aprovação:
- $MS \geq 7,0 \Rightarrow$ Aprovado direto
- $MS \geq 4,0$ e $MS < 7,0 \Rightarrow$ Prova Final
- $MS < 4,0 \Rightarrow$ Reprovado direto
- PROVA FINAL

$PF = 10 - MS \geq 5,0$ para ser aprovado

Programação se aprende programando.

Compilador C

Fique à vontade para usar qualquer compilador C.
No curso, os programas serão apresentados via o DEV C++



<https://sourceforge.net/projects/orwelldevcpp/>



https://www.onlinegdb.com/online_c_compiler

Introdução à Linguagem C

- Criada em 1972 por Dennis Ritchie que também é um dos criadores do Unix no AT&T Bell Labs
- Criada para desenvolver o sistema operacional Unix (escrito originalmente em *Assembly*)
- A linguagem começou a se tornar popular depois do lançamento do primeiro livro
- Na década de 80 foi adaptado para uso em IBM PC
- Nesta época surgiu também o C++ (OO)
- Em 1983 o American National Standards Institute (ANSI) formou um comitê para estabelecer um padrão para a linguagem – finalizado em 1989

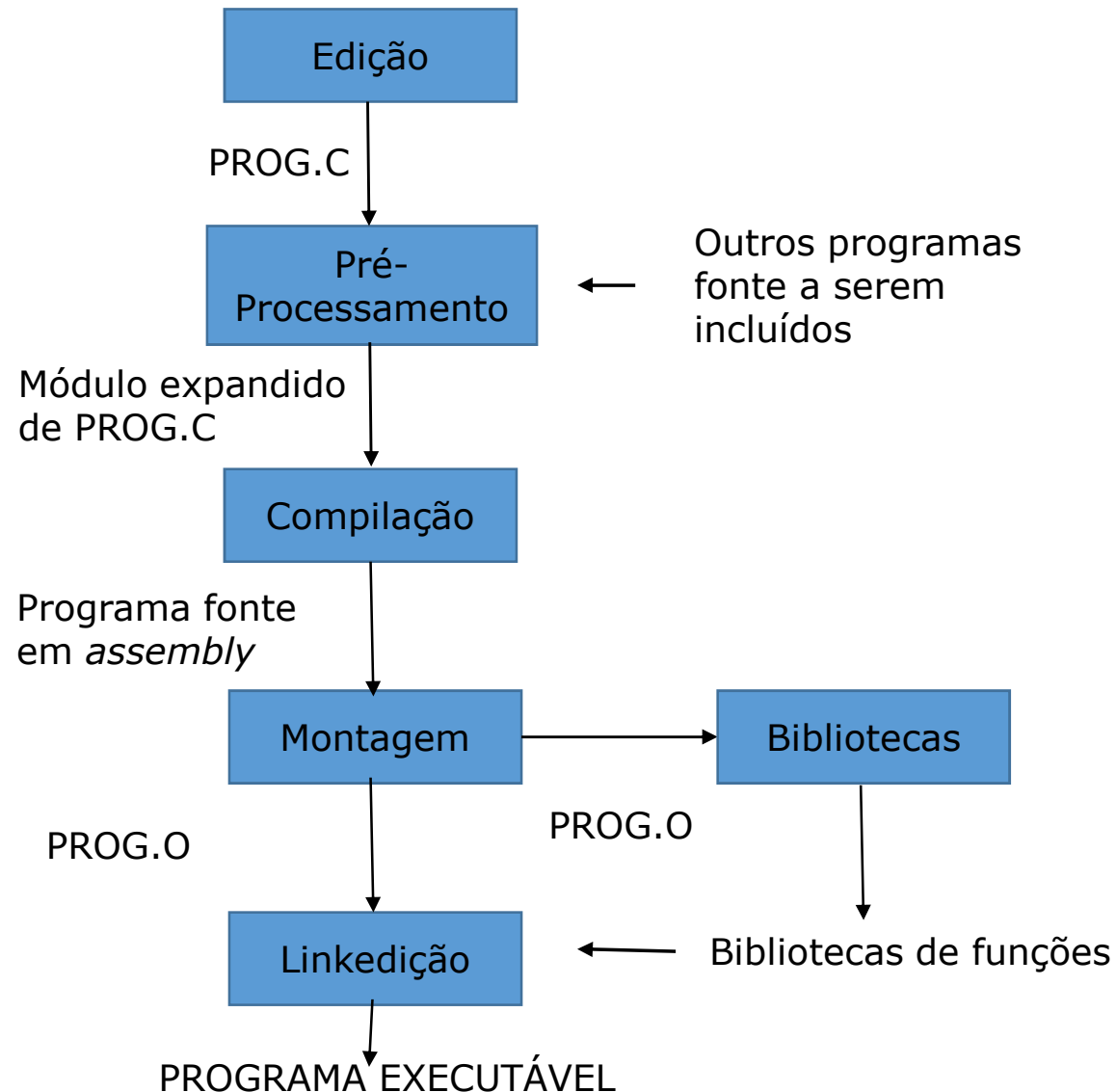
Introdução à Linguagem C

- Linguagem de alto nível
- Procedural
- Tipagem estática (pré-declaração) e forte (fixo)
- Permite uma otimização de código normalmente alcançável em programas escritos em *assembler*
- Baixos requerimentos de hardware
- Possibilita reaproveitamento de código
- *Case sensitive*

Introdução à Linguagem C

- Linguagem de Programação de uso geral.
- Principais características:
 - Alto grau de portabilidade
 - Uso geral
 - Gera um código executável (objeto) compacto e rápido
 - Expressões compactas
 - Grande conjunto de operadores
 - Poderosas estruturas de dados
 - Mecanismos de controle de fluxo eficientes

Criação de um Código Executável em C



Estrutura de um programa em C

```
[< definições de pré-processamento>]
[<declaração de variáveis globais>]
[<declaração de protótipos >]
main () /* pode ser "void main ()" ou "int main ()" */
{
    /* corpo da função principal com declarações de suas variáveis, seus
    comandos e funções.
    */
}
[
[ <tipo> função ([<lista de parâmetros>])
[ <declaração de parâmetros> ]
{
    /* corpo da função com suas declarações de variáveis,
    comandos e funções.
    */
}
]
```

Primeiro Programa

```
#include <stdio.h> /* diretiva de inclusão da biblioteca padrão */
int main()          /* função principal */
{
    printf("Alo Mundo!\n"); /* função impressão */

    return 0;
}
```

Saída de Dados

Função printf ()

```
printf ("expressão de controle", arg1, arg2, ...)
```

Ex.:

```
int main ()  
{  
    printf ("Esta é a aula numero %d (um)", 1);  
    return 0;  
}
```

Saída de Dados - printf (continuação)

- Imprime a expressão de controle na tela
- Os códigos de formatação (iniciados por %) devem ser colocados para a impressão dos parâmetros arg1, arg2...
- Deve haver um argumento (arg1, arg2...) para cada código de formatação

Principais códigos de formatação:

- %u – inteiro decimal sem sinal
- %c – caracter simples
- %d – inteiro decimal com sinal
- %e – real em notação científica
- %f – real em ponto flutuante
- %s – cadeia de caracteres
- %x – inteiro com base hexadecimal
- %o – inteiro com base octal

Saída de Dados - printf (continuação)

Imprimindo uma cadeia de caracteres

```
int main ()
{
    printf ("%s  - Universidade do Estado do Rio de Janeiro\n", "UERJ");
    printf ("Linguagem de Programacao");
    return 0;
}
```

Caracteres de impressão que não são obtidos diretamente do teclado:

- \n – nova linha
- \t – tabulação
- \b – *backspace*
- \f – salta uma página de formulário
- \0 – caracter nulo (terminador de *string*)

Formatação dos resultados de saída

Para formatar os resultados de uma forma organizada em colunas, com títulos e legendas é preciso definir o tamanho dos campos onde os dados serão escritos. É utilizado em conjunto com o código de formatação de cada argumento.

Caso geral: %[-][tamanho][.precisão]{d,o,x,u,c,s,e,f}

Ex.:

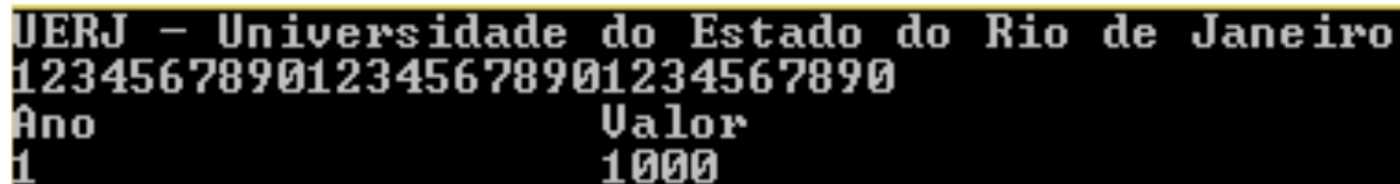
```
printf ("%s - Universidade do Estado do Rio de Janeiro\n", "UERJ");  
printf ("123456789012345678901234567890\n");  
printf ("%10s%10c%10s\n", "Ano", ' ', "Valor");  
printf ("%9d%11c%10d\n", 1, ' ', 1000);
```

```
UERJ - Universidade do Estado do Rio de Janeiro  
123456789012345678901234567890  
      Ano      Valor  
      1      1000
```

Formatação dos resultados de saída (continuação)

O sinal de menos [-] precedendo a especificação do tamanho do campo justifica os campos à esquerda.

```
printf ("%s - Universidade do Estado do Rio de Janeiro\n", "UERJ");  
printf ("123456789012345678901234567890\n");  
printf ("%s%-10s%-10c%-10s\n", "Ano", ' ', "Valor");  
printf ("%s%-9d%-11c%-10d\n", 1, ' ', 1000);
```



```
UERJ - Universidade do Estado do Rio de Janeiro  
123456789012345678901234567890  
Ano          Valor  
1            1000
```

Saída de Dados – puts () e putchar() (continuação)

- puts () – imprime uma string e ao final pula uma linha

Os comandos abaixo são similares:

```
puts ("string");  
printf ("%s\n", "string");
```

- putchar() – imprime um único caracter mas sem pular a linha

Os comandos abaixo são similares:

```
putchar ('c');  
printf ("%c", 'c');
```


Saída de Dados - Exercício

Escrever um programa que imprima na tela:

123456789012345678901234567890

linha 1

linha 2

linha 3

```
int main ()
```

```
{
```

```
    puts ("123456789012345678901234567890");
```

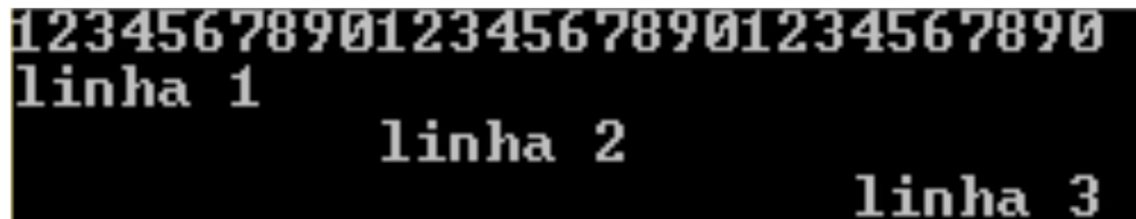
```
    printf ("%5s%2d%\n", "linha", 1);
```

```
    printf ("%10c%-5s%2d\n", ' ', "linha", 2);
```

```
    printf ("%23c%-5s%2d\n", ' ', "linha", 3);
```

```
    return 0;
```

```
}
```



```
123456789012345678901234567890
linha 1
                linha 2
                                linha 3
```