

Introdução a Programação de Computadores usando a Linguagem C

C3

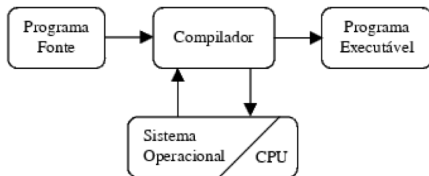
Universidade Federal de Rio Grande
Centro de Ciências Computacionais

Linguagem de Programação

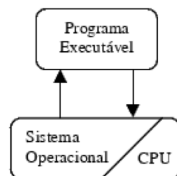
- ▶ Humanos: Linguagem Natural
- ▶ Computadores: Linguagem de Máquina
- ▶ Como se comunicar? Linguagem de programação.
- ▶ Programa para traduzir linguagem Natural para Linguagem de Máquina: Compilador ou Interpretador

Compilador

- ▶ Traduz todo o código fonte em linguagem de máquina.
- ▶ Gera um arquivo chamado com o código objeto ou “executável”.
- ▶ Em um linguagem compilada, o arquivo executável não necessita mais do compilador para executar.
- ▶ Linguagens Compiladas: C, C++, Pascal, Ada, Fortran, ...



GERAÇÃO DO PROGRAMA EXECUTÁVEL



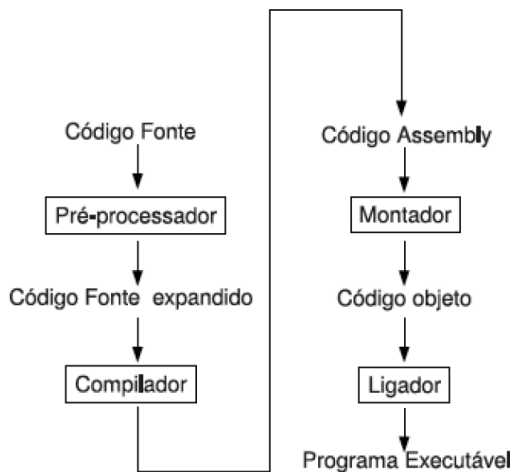
EXECUÇÃO DO PROGRAMA

Linguagens Híbridas

- ▶ Existem linguagens que são híbridas.
- ▶ Exemplo: Java
- ▶ Um código fonte escrito em java primeiro deve ser compilado.
 - ▶ Gerando um arquivo .class que tem um código denominado bytecode.
 - ▶ O bytecode não é nem linguagem natural, nem linguagem de máquina. É um código intermediário.
- ▶ Depois o arquivo .class (com o bytecode) deve ser interpretado, e então executado, pela Máquina Virtual Java (JVM-Java Virtual Machine).

Compilador Linguagem C

► Etapas do Compilador C



Compilador Linguagem C

- ▶ **Código Fonte:** arquivo texto que contém um programa escrito na Linguagem C.
- ▶ **Pré-processador:** Etapa do compilador que prepara o código fonte para ser compilado. Processa macros como o `#define`. Gera o código-expandido.
- ▶ **Compilador:** traduz o código-expandido em Linguagem Assembly.
 - ▶ **Linguagem Assembly ou Linguagem de Montagem:** é uma linguagem de máquina com notação legível para seres humanos (ADD, CMP, MOV, ...).
- ▶ **Montador ou Assembler:** traduz o código em Assembly para linguagem de máquina (binário puro) gerando o código objeto.
- ▶ **Ligador ou Linker:** “junta” o código objeto com as funções de outras bibliotecas para tornar o código executável.

Linguagem C - Histórico

- ▶ Algol 60 (1960): Criada para substituir o FORTRAN. Muito alto nível.
- ▶ CPL (1967): Londres/Cambridge, muito difícil de programar.
- ▶ BCPL (1967): Tentativa de aproveitar melhores funções da CPL.
- ▶ B (1970): Ken Thompson (UNIX/PDP11 na Bell Lab.) Linguagem limitada (baseada na CPL).
- ▶ C (1972): Dennis Ritchie (Bell Lab.) Contato com o computador real. Incluída na dist. UNIX/PDP11.
- ▶ ANSI C (1985): American National Standards Institute. Padrão oficial de C.

Linguagem C - Características Básicas

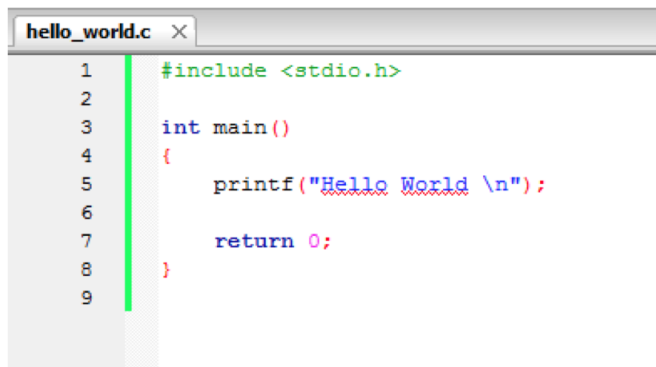
- ▶ Código compilado.
- ▶ Linguagem simples (poucas instruções).
- ▶ Linguagem estruturada → funções.
- ▶ Linguagem tipada.
- ▶ Case-Sensitive: letras minúsculas são diferentes de letras maiúsculas.
- ▶ $C \neq C++$

Que programa eu uso para fazer um programa em C

- ▶ O código fonte pode ser escrito em qualquer editor de texto (bloco de notas, kedit, notepad++, ...) e deve ser salvo com a extensão `.c`
 - ▶ editor de texto \neq processador de texto (MS-Word, ...)
- ▶ Depois usar um compilador para transformar o arquivo `.c` no executável.
- ▶ Compiladores:
 - ▶ Linux: gcc
 - ▶ Windows: MinGW
 - ▶ ...
- ▶ Se preferir, você pode usar uma IDE (Integrated Development Environment):
 - ▶ Code::Blocks
 - ▶ NetBeans
 - ▶ Eclipse
 - ▶ MS Visual Studio
 - ▶ XCode
 - ▶ ...

Criando compilando um programa em C - Terminal

Digitar o código-fonte



```
hello_world.c ×  
1  #include <stdio.h>  
2  
3  int main()  
4  {  
5      printf("Hello World \n");  
6  
7      return 0;  
8  }  
9
```

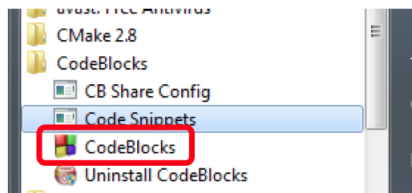
Criando compilando um programa em C - Terminal

- Abrir o terminal
- Alterar o diretório para onde se encontra o arquivo C
 - Usar o comando ***cd***
- Compilar o arquivo fonte
 - **`gcc -o meu_arquivo meu_arquivo.c`**
- Executando o arquivo compilado
 - **`./meu_arquivo`**

Criando compilando um programa em C - Code::Blocks

- No Code::Blocks (Windows e Linux)

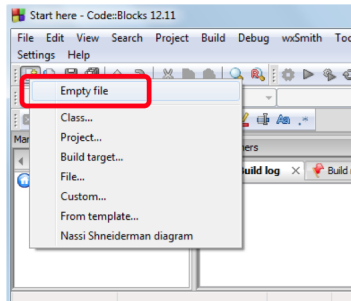
1) Iniciar o programa



Criando compilando um programa em C - Code::Blocks

2) Criar um arquivo em branco

-> File > New > Empty File (ctrl+shift+N)

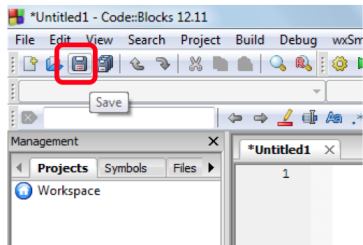


Criando compilando um programa em C - Code::Blocks

3) Salvar o arquivo com extensão .c

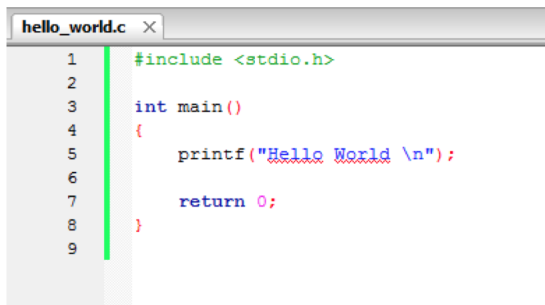
-> File > Save File (ctrl+S)

Obs.: evitar o uso de caracteres especiais e espaços no nome



Criando compilando um programa em C - Code::Blocks

4) Digitar o código-fonte



The screenshot shows a Code::Blocks editor window with a single tab titled 'hello_world.c'. The editor contains the following C code:

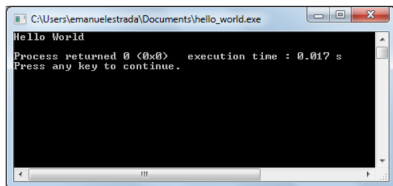
```
1  #include <stdio.h>
2
3  int main()
4  {
5      printf("Hello World \n");
6
7      return 0;
8  }
9
```

The code is color-coded: preprocessor directives are green, keywords are blue, string literals are red, and identifiers are black. A vertical green line is positioned at the start of line 5.

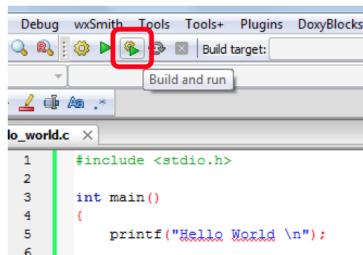
Criando compilando um programa em C - Code::Blocks

5) Compilar e executar

-> Build > Build and Run (F9)



```
CAUsers\emanuelestrada\Documents\hello_world.exe
Hello World
Process returned 0 (0x0)   execution time : 0.017 s
Press any key to continue.
```



Alô Mundo!

```
#include<stdio.h>

int main() {

    printf("Alo Mundo!! \n");

    return 0;
}
```

Alô Mundo!

```
/** Decl. Bibliotecas *****/  
#include<stdio.h>  
  
/** Programa principal *****/  
int main() {  
  
    /* Instrucoes */  
    printf("Alo Mundo!! \n");  
  
    /* Retorna o status do programa  
       para o Sistema Operacional */  
    return 0;  
}
```

Programas em C

- ▶ Um programa em C é uma coleção de funções.
- ▶ A função principal é a `main`. É a primeira função a ser executada.
- ▶ Case-Sensitive:
 - ▶ `int` \neq `Int` \neq `INT`
 - ▶ `main` \neq `Main` \neq `MAIN`
- ▶ Um bloco de instruções é limitado por `{` e `}`
- ▶ Toda instrução em C termina com;
- ▶ O compilador C ignora espaços e linhas em branco entre palavras e símbolos.

Comentários

- ▶ Quando fazemos um programa, uma boa idéia é usar comentários que ajudem a elucidar o funcionamento do mesmo.
- ▶ Para comentar o código usamos: `/* e */`.
- ▶ O compilador C desconsidera qualquer texto que esteja entre `/* e */`.
- ▶ Um comentário pode, inclusive, ter mais de uma linha.

```
#include<stdio.h>
/* Meu primeiro
   Programa */
int main() {
    /* Imprime Mensagem na Tela */
    printf("Alo Mundo!!");
    return 0;
}
```

Palavras Reservadas em C

auto	else	signed
break	enum	sizeof
case	float	struct
char	for	switch
const	if	typedef
default	int	unsigned
do	long	void
double	return	while

Tipos de Dados

Tipo	Tamanho (bytes)	Tamanho (bits)	Minimo	Maximo
char	1	8	-128 (-2^7)	127 (2^7-1)
unsigned char	1	8	0	255 ($2^8 - 1$)
short int	2	16	-32768 (-2^{15})	32767 ($2^{15}-1$)
unsigned short int ...	2	16	0	65535 ($2^{16}-1$)
int	4	32	-2147483648 (-2^{31})	2147483647 ($2^{31}-1$)
unsigned int	4	32	0	4294967295 ($2^{32}-1$)
long int	8	64	-9223372036854775808 (-2^{63})	9223372036854775807 ($2^{63}-1$)
unsigned long int ...	8	64	0	18446744073709551615 ($2^{64}-1$)
float.....	4	32	1.175494e-38	3.402823e+38
double.....	8	64	2.225074e-308	1.797693e+308
long double.....	16	128	3.362103e-4932	1.189731e+4932

Declaração de Variáveis

- ▶ Uma variável é uma posição nomeada de memória que é usada para guardar um valor que pode ser modificado pelo programa.
- ▶ Todas as variáveis em C devem ser declaradas antes de serem utilizadas.
- ▶ As variáveis devem ser declaradas no início da função.
- ▶ Forma Geral:

```
tipo lista_de_variaveis;
```

- ▶ Onde, tipo deve ser um tipo de dado válido em C.
- ▶ lista_de_variáveis pode consistir em um ou mais nomes de identificadores separados por vírgula.

Declaração de Variáveis

► Exemplo:

```
int i, j, m;  
short int si;  
unsigned int ui;  
double balance, profit, loss;
```

Operador de Atribuição

- ▶ O operador de atribuição em C é o `=`.

```
int x, y;  
x=1;  
y=2;
```

- ▶ Também é possível fazer atribuições múltiplas.Exemplo:

```
int x, y, z;  
  
/* atribuindo para todas as variaveis o valor 0*/  
x = y = z = 0;
```

Inicialização de Variáveis

- ▶ Pode-se atribuir valores a variáveis na linguagem C no momento em que elas são declaradas, colocando um sinal de igual e uma constante após o nome da variável.
- ▶ Forma Geral:

```
tipo nome_da_variavel = constante;
```

- ▶ Exemplo:

```
int first = 0;  
float balance = 123.23;
```

Tipo char

- ▶ Os caracteres são um tipo de dado: o char
- ▶ O C trata os caracteres ('a', 'b', 'x', ...) como sendo variáveis de um byte (8 bits).

```
char Ch;  
char letra;  
Ch='D';  
letra='a';
```

- ▶ Na linguagem C, também podemos usar o tipo char para armazenar valores numéricos inteiros (tamanho 1 byte).
- ▶ Tabela ASCII

```
char c;  
c=65;  /* c='A'; */
```

Operadores Aritméticos

Operador	Ação
-	Subtração
+	Adição
*	Multiplicação
/	Divisão
%	Módulo da divisão (resto)
--	Decremento
++	Incremento

Incremento e Decremento

- ▶ $x = x + 1$ é o mesmo que $++x$ ou $x++$.
- ▶ $x = x - 1$ é o mesmo que $--x$ ou $x--$.
- ▶ Porém:
 - ▶
 - ▶ $x = 10; y = ++x;$ é diferente de $x = 10; y = x++;$
 - ▶ No primeiro caso, o x é somado e depois atribuído e, no segundo caso, o x é atribuído e depois somado - em ambos os casos, x fica com 11)

Saída: printf

- ▶ Para mostrar/imprimir uma mensagem na tela usamos a função `printf`.
- ▶ A função `printf` faz parte da biblioteca `stdio` (*Standard Input Output*).
- ▶ A função `printf` tem a seguinte forma geral:

`printf(String_de_Controle, Lista_de_Variáveis);`

- ▶ **String de Controle:** uma descrição de tudo que a função vai mostrar na tela.
- ▶ **Lista de Variáveis:** variáveis que serão mostradas separadas por `,`

Saída: printf

- ▶ A string de controle mostra os caracteres que devem ser colocados na tela e as variáveis.
- ▶ Na string de controle indicamos qual tipo e em que posição estão as variáveis a serem apresentadas usando códigos de controle (%).
- ▶ É muito importante que, para cada código de controle, tenhamos uma variável na lista de argumentos.
- ▶ Principais códigos de controle:

Código	Tipo
%d	int
%f	float
%c	char
%s	String
%%	Coloca na tela um %

Saída: printf

- Vamos ver alguns exemplos de printf e o que eles exibem:

```
printf("Teste %% %%");  
// "Teste % %"
```

```
printf("%f", 40.345);  
//"40.345"
```

```
printf("Um caractere %c e um inteiro %d",'D',120);  
//"Um caractere D e um inteiro 120"
```

```
printf("%s e um exemplo","Este");  
//"Este e um exemplo"
```

```
printf("%s%d%%","Juros de ",10);  
//"Juros de 10%"
```

Saída: printf

- ▶ Para pular uma linha usamos o caractere especial: `\n`
- ▶ Exemplo:

```
printf("1 1 1 \n");  
printf("2 2 2 2 \n 3 3");  
printf("\n 4 4 4 \n");  
printf("\n \n");  
printf("7 \n");
```

- ▶ Saída:

```
1 1 1  
2 2 2 2  
 3 3  
 4 4 4  
  
7
```

Saída: printf

- Principais caracteres especiais:

Código	Caractere
<code>\n</code>	quebra linha
<code>\t</code>	tab
<code>\"</code>	aspas duplas
<code>'</code>	aspas simples
<code>\\</code>	<code>\</code>
<code>\a</code>	beep

Entrada: scanf

- ▶ Para ler um valor do teclado usamos a função `scanf`.
- ▶ A função `scanf` faz parte da biblioteca `stdio` (*Standard Input Output*).
- ▶ A função `scanf` tem a seguinte forma geral:

`scanf(String_de_Controle, Lista_de_Enderecos);`

- ▶ **String de Controle:** uma descrição do tipo variável (código de controle % - igual ao `printf`).
- ▶ **Lista de Endereços:** endereços de memória onde os valores lidos serão armazenados.
 - ▶ Se não estivermos trabalhando com ponteiros a forma será:
`&NomeVariável`

Entrada: scanf

- Vamos ver alguns exemplos de scanf:

```
int soma;  
float media;  
  
//Le um valor inteiro e guarda em soma  
scanf("%d",&soma);  
  
//Le um valor real e guarda em media  
scanf("%f",&media);
```

Referências

- ▶ ASCENCIO, Ana F. G.; CAMPOS, Edilene A. V. Fundamentos da programa de computadores. Pearson Prentice Hall, 2007. (Bom para a parte de Lógica).
- ▶ Schildt, Herbert. C completo e total. Pearson Makron Books, 1997.
- ▶ Notas de aula do Prof. Flavio Keidi Miyazawa.
- ▶ Notas de aula do Prof. Emanuel Estrada.
- ▶ Notas de aula do Prof. Alessandro Bicho.