




Disciplina:

# Programação Computacional

Prof. Fernando Rodrigues  
e-mail: fernandorodrigues@sobral.ufc.br

## Aula 08: Conceitos básicos da linguagem C

- ❖ Principais bibliotecas
  - ❖ Tipos de dados
  - ❖ Palavras reservadas
  - ❖ Operadores
  - ❖ Atribuição
  - ❖ Funções de Entrada e Saída (printf() e scanf())
- 

# Conceitos básicos da linguagem C

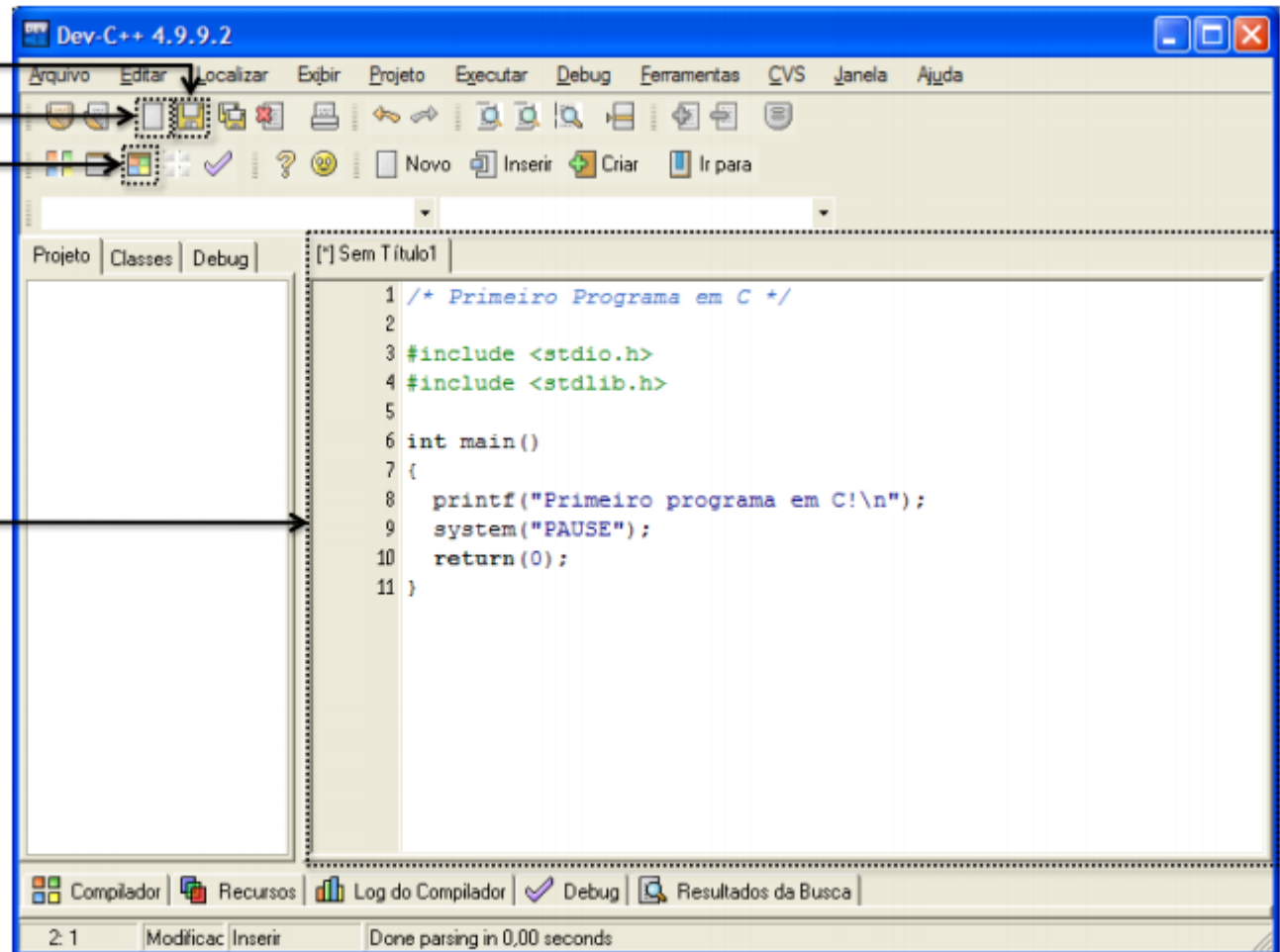
# Ambiente de Programação: Dev C++

Salvar programa-fonte

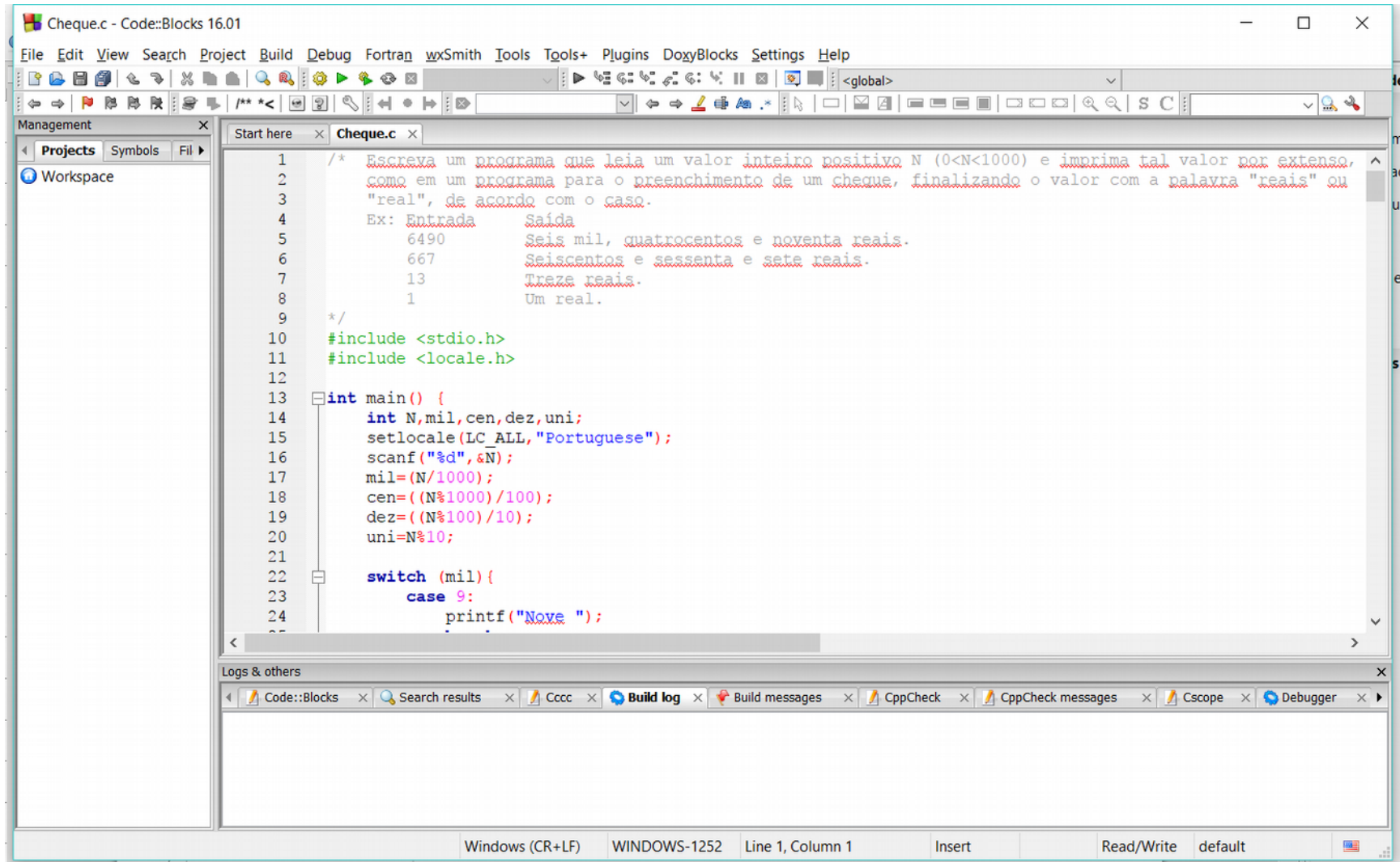
Novo programa-fonte

Compilar e executar

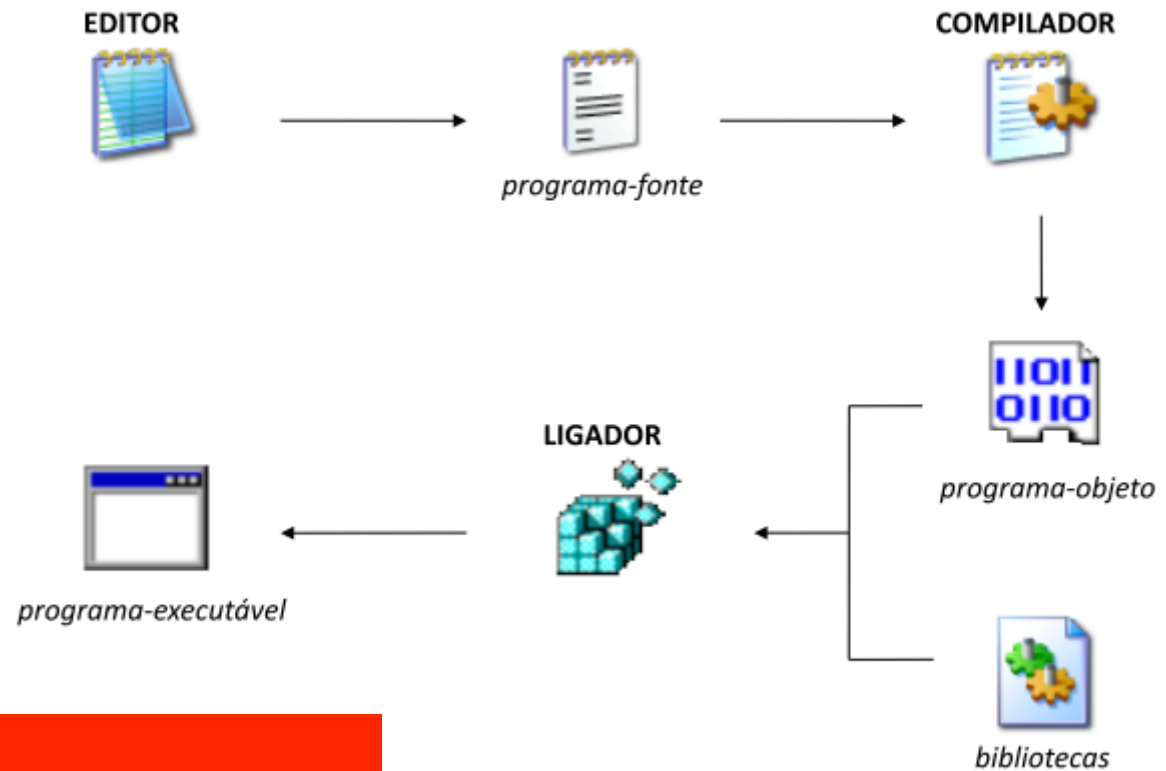
Janela de edição



# Ambiente de Programação: CodeBlocks



# Criação de programa



**Lembre-se sempre:**

Só se aprende a programar programando!  
A prática é fundamental.

# O menor programa em C

---

Todo programa escrito na linguagem C possui uma estrutura básica, sobre a qual são adicionadas as instruções e comandos que se deseja que o programa execute.

```
main()  
{  
  
}
```

## A função `main()`

- É a função principal de um programa em C.
- **Sempre** deverá existir.
- Marca o início da execução do programa.

# Informações importantes

---

## ► Tudo deve ter começo e fim

- Os símbolos `/*` e `*/` indicam o começo e o fim de um comentário.
- Os símbolos `{` e `}` indicam o começo e o fim de um bloco de comandos.
- Os símbolos `"` e `"` indicam o começo e o fim de uma cadeia de caracteres.
- Todo comando deve ser encerrado com o símbolo `;` (ponto-e-vírgula).

```
/* Programa Hello, world! */  
  
#include <stdio.h>  
  
main()  
{  
    printf("Hello, world!\n");  
}
```

# Tipos primitivos

A linguagem C realiza operações sobre dados **numéricos** e **não-numéricos**.

Os dados numéricos podem ser de 3 tipos distintos:

**int** : para representar valores inteiros;

**float** : para representar valores de ponto flutuante;

**double** : para representar valores de ponto flutuante de precisão dupla.

O tipo **char** permite manipular elementos não-numéricos (dados como letras, dígitos ou outro símbolo gráfico).

O tipo **void** é utilizado apenas para funções (que não retornam valores) e ponteiros genéricos (como veremos posteriormente).

A linguagem C não permite que se declare uma variável do tipo **void**. Esse tipo de dados só deve ser usado para declarar funções que não retornam valor ou ponteiro genérico.

A linguagem C ANSI não possui o tipo **boolean**. O mesmo é definido apenas para a linguagem C++ (como tipo “**bool**”).



# Modificadores de tipos de dados

---

- **signed:**
  - determina que uma variável declarada dos tipos *char* ou *int* poderá ter valores positivos ou negativos. Trata-se do modo-padrão de definição de variáveis desses tipos, e, por esse motivo, raramente é usado.
- **unsigned:**
  - determina que uma variável declarada dos tipos *char* ou *int* somente poderá ter valores positivos e o valor zero. Nesse caso, a variável perde seu bit de sinal, o que dobra a sua capacidade de armazenamento para valores positivos.
- **short**
- **long**

# Modificadores de tipos de dados

---

- `signed`
- `unsigned`
- `short`:
  - O modificador *short* determina que uma variável do tipo *int* terá apenas 16 bits (inteiro pequeno), independentemente do processador.
- `long`:
  - O modificador *long* determina que uma variável do tipo *int* terá 32 bits (inteiro grande), independentemente do processador. Também determina que o tipo *double* possua maior precisão.



A linguagem C permite que se utilize mais de um modificador de tipo sobre um mesmo tipo.

# Modificadores de tipos (combinações)

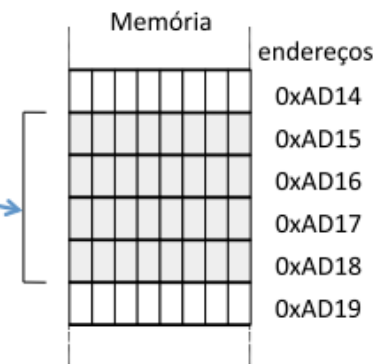
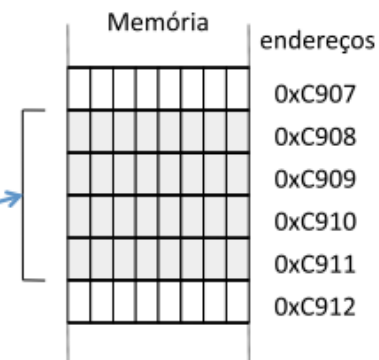
Tipo	Bits	Intervalo de valores
char	8	-128 A 127
unsigned char	8	0 A 255
signed char	8	-128 A 127
int	32	-2.147.483.648 A 2.147.483.647
unsigned int	32	0 A 4.294.967.295
signed int	32	-32.768 A 32.767
short int	16	-32.768 A 32.767
unsigned short int	16	0 A 65.535
signed short int	16	-32.768 A 32.767
long int	32	-2.147.483.648 A 2.147.483.647
unsigned long int	32	0 A 4.294.967.295
signed long int	32	-2.147.483.648 A 2.147.483.647
float	32	1,175494E-038 A 3,402823E+038
double	64	2,225074E-308 A 1,797693E+308
long double	96	3,4E-4932 A 3,4E+4932

# Declaração de variáveis e/ou constantes

- Sintaxe: **tipo** **nome\_variavel 1, nome\_variavel 2;**

A declaração é um comando que associa o nome de uma constante ou de uma variável a um determinado endereço de memória.

```
main()
{
    const int i = 5;
    int j;
    // comandos
}
```



# Declarando variáveis e/ou constantes

---

## Declaração prévia

Toda constante ou variável deve ser declarada **antes** de ser usada.

## Declaração única

O nome da constante ou variável deve ser **único**.

# Identificadores

---

Os nomes dados às constantes ou variáveis de um programa denominam-se **identificadores**. Para definir um identificador pode-se utilizar uma combinação de:

- letras (maiúsculas e minúsculas, sem acentuação).
- dígitos (0 a 9), desde que o primeiro caractere não seja um dígito.
- caractere '\_' (sublinha).

A linguagem C diferencia letras maiúsculas e minúsculas.

```
int    F;           // uma variável inteira denominada F
float  f;           // uma variável real denominada f
double valor de A;  // identificador inválido!
char   la_inicial;  // identificador inválido!
```

# Constantes x Variáveis

---

Em linguagens de programação de alto nível, os dados podem ser de duas naturezas: **constantes** ou **variáveis**.

## Constantes

Assumem um único valor, do início ao fim da execução do programa, não podendo ser alterado por nenhum comando ou função.

## Variáveis

Assumem um valor inicial, que pode ser alterado por alguma função ou comando durante a execução do programa.

# Características da Linguagem C

---

- Case sensitive;
- Fortemente tipada;
- Compilada para código nativo;
- Grande disponibilidade de bibliotecas de funções;
- Padrão seguido por várias outras linguagens (C-Like): C++, Java, C#, Python;
- Alta disponibilidade de compiladores para vários ambientes;
- Filosofia da linguagem:
  - “O programador **sabe** o que está fazendo”!





# Palavras reservadas

---

As seguintes palavras não podem ser utilizadas para denominar entidades (constantes, variáveis, estruturas, funções etc.) criadas pelo programador:

- Armazenamento: **auto, extern, register, static.**
- Tipagem: **char, const, double, enum, float, int, long, short, signed, struct, typedef, union, unsigned, void, volatile.**
- Controle de execução: **break, continue, goto, return.**
- Comandos de seleção: **case, default, else, if, switch.**
- Comandos de iteração: **do, for, while.**
- Função: **sizeof( ).**

# Principais bibliotecas

---

A linguagem C possui um conjunto mínimo de instruções, visando a criação de programas executáveis de tamanho pequeno. A adição de novas funcionalidades é feita através da inclusão de bibliotecas, que contêm classes de funções específicas para o tratamento de dados desejado.

```
#include <stdio.h>    // biblioteca de funções de entrada e saída
#include <stdlib.h>    // biblioteca de funções do sistema operacional

main()
{
    printf("Primeiro programa em C!\n");
    system("PAUSE");
}
```

A inclusão de bibliotecas devem ser as primeiras instruções de um programa em C.

# Principais bibliotecas

---

Biblioteca	Principais funcionalidades
<code>stdio.h</code>	entrada e saída de dados.
<code>stdlib.h</code>	alocação de memória e comandos para o sistema operacional.
<code>math.h</code>	funções matemáticas.
<code>time.h</code>	manipulação de dados nos formatos de data e hora.
<code>ctype.h</code>	manipulação de caracteres.
<code>string.h</code>	manipulação de cadeias de caracteres.
<code>conio.h</code>	manipulação do cursor na tela.

# Atribuição

A atribuição é um comando utilizado para modificar o valor de uma variável.

```
main()
{
    int    i;
    float  x;
    char   c;

    i = 5;      // a variável i recebe o valor inteiro 5.
    x = 5.0;    // a variável x recebe o valor "real" 5.
    c = '5';    // a variável c recebe o caractere 5.
    ...
}
```

## Atenção

- o símbolo de atribuição = não significa igualdade.
- a atribuição sempre atua da direita para a esquerda (←).

# Atribuição

---

A atribuição é um comando destrutivo, ou seja, o valor anteriormente armazenado pela variável à esquerda do símbolo `=` será substituído pelo valor da constante, variável ou expressão no lado direito.

```
main()  
{  
    int i;  
    int j;  
  
    i = 2;           // i recebe o valor 2.  
    j = i;           // j recebe o valor de i.  
    i = 0;           // i recebe o valor 0.  
    i = i + 1;       // some 1 ao valor atual de i e armazene o resultado em i.  
    i = j;           // i recebe o valor de j.  
  
}
```

# Operadores

Os seguintes símbolos são utilizados como operadores na linguagem C:

Aritméticos	
Símbolo	Operação
+	adição
—	subtração
*	multiplicação
/	divisão
%	módulo

Relacionais	
Símbolo	Significado
<	menor que
>	maior que
<=	menor ou igual à
>=	maior ou igual à
==	igual
!=	diferente

Lógicos	
Símbolo	Operação
&&	AND
	OR
!	NOT

# Operadores Aritméticos

---

A precedência das operações aritméticas em C obedece às regras estabelecidas pela Álgebra. Os operadores com mesmo nível de precedência são avaliados pelo compilador da esquerda para a direita.

Símbolo	Operação	Resultado	Precedência
+	adição	soma dos argumentos	baixa
-	subtração	diferença dos argumentos	baixa
*	multiplicação	produto dos argumentos	média
/	divisão	quociente dos argumentos	média
%	módulo	resto da divisão inteira	média
++	incremento	adiciona 1 ao operando	alta
--	decremento	subtrai 1 do operando	alta

A precedência das operações pode ser modificada com o uso de parênteses.