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.

```c
#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
`stdio.h`	entrada e saída de dados.
`stdlib.h`	alocação de memória e comandos para o sistema operacional.
`math.h`	funções matemáticas.
`time.h`	manipulação de dados nos formatos de data e hora.
`ctype.h`	manipulação de caracteres.
`string.h`	manipulação de cadeias de caracteres.
`conio.h`	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 caratere 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.

Operadores Lógicos

Utilizados para testar mais de uma condição, simultaneamente.

Operador lógico	Significado	Precedência
!	NOT	altíssima
&&	AND	alta
||	OR	baixa

Operadores lógicos tem precedência mais baixa que os operadores relacionais (exceto o !).

&&		expressão 1	
		1	0
expressão 2	1	1	0
	0	0	0

||		expressão 1	
		1	0
expressão 2	1	1	1
	0	1	0

expressão	!expressão
1	0
0	1

Operadores (Exemplo)

Calcular o perímetro e o volume de uma esfera de raio r = 3.

$$P = 4\pi r^2.$$

$$V = \frac{4}{3}\pi r^3$$

```
main()
{
  const float PI = 3.14159;
  float r = 3;
  float p, v;

  p = 4*PI*r*r;
  v = (4.0/3.0)*PI*r*r*r;

}
```

Operadores de Atribuição

- Suponha: int c = 3, d = 5, e = 4, f = 6, g = 12

c += 7 => c = c + 7 => ?

d -= 4 => d = d - 4 => ?

e *= 5 => e = e * 5 => ?

f /= 3 => f = f / 3 => ?

g %= 9 => g = g % 9 => ?

Incremento e decremento

São utilizados para adicionar ou subtrair 1 unidade de uma variável inteira.

```
i++;        // equivale ao comando i = i + 1;
j--;        // equivale ao comando j = j – 1;
```

Notação pré-fixa

O valor da variável é atualizado antes de ser utilizado na expressão.

```
i = 3;
j = ++i;    // i assume o valor 4, j assume o valor de i
```

Notação pós-fixa

O valor da variável é atualizado depois de ser utilizado na expressão.

```
i = 3;
j = i++;    // j assume o valor de i, i assume o valor 4
```

Operadores de Incremento e Decremento		
pré-incremento	++a	Incrementa **a** por **1**, depois utiliza o novo valor de **a** na expressão em que **a** reside.
pós-incremento	a++	Utiliza o valor atual de **a** na expressão em que **a** reside, depois incrementa **a** por **1**.
pré-decremento	--b	Decrementa **b** por **1**, depois utiliza o novo valor de **b** na expressão em que **b** reside.
pós-decremento	b--	Utiliza o valor atual de **b** na expressão em que **b** reside, depois incrementa **b** por **1**.

```
int c = 5;
printf(c++);   → 5
printf(++c);   → 6
```

Prioridades de operadores

- 2*4+2 = ?
- 2*(4+2) = ?

- 3*4/2 = ?
- 3*(4/2) = ?

Função de escrita Printf()

A função `printf()` é utilizada para exibição de informações. Sua sintaxe é:

```
printf("expressão de controle", lista de argumentos);
```

A **`"expressão de controle"`** contém a mensagem que será exibida na tela, juntamente com os caracteres especiais de exibição e os códigos de formatação dos argumentos.

A **`lista de argumentos`** corresponde à constantes, variáveis e expressões que serão exibidas na tela, de acordo com os formatos estabelecidos pela `"expressão de controle"`.

Símbolos utilizados na função printf()

Servem para controle e formatação da exibição em tela.

Caractere	Ação
\n	nova linha
\t	tabulação
\b	retrocesso (*backspace*)
\f	novo formulário
\a	alerta (sinal sonoro)
\r	início da linha
\0	caractere nulo
\"	exibe o caractere "
\\	exibe o caractere \

Código	Exibição
%c	caractere simples
%s	cadeia de caracteres
%d	valor inteiro
%u	valor inteiro sem sinal
%f	valor de ponto flutuante
%e	notação científica
%o	valor octal
%x	valor hexadecimal
%%	caractere %

Obs: Para valores “double”, utilizar o código de formatação %lf (long float)

printf( ) - Exemplos

Exibir uma mensagem:

```
printf("Bom dia!");
```

Exibir uma mensagem e pular duas linhas:

```
printf("Bom dia!\n\n");
```

Exibir o valor de uma variável inteira:

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

Exibir o valor de uma variável inteira e uma variável real:

```
printf("%d %f", j, x);
```

Exibir mensagens e valores de variáveis:

```
printf("Valor de j = %d\nValor de x = %f\n", j, x);
```

Função de leitura scanf()

A função `scanf()` é utilizada para leitura de dados pelo teclado. Sua sintaxe é:

```
scanf("expressão de controle", lista de argumentos);
```

Diferentemente da função `printf()`, a **`"expressão de controle"`** da função `scanf()` deverá conter apenas os códigos de formatação das variáveis a serem lidas.

A **`lista de argumentos`** é composta pelos nomes das variáveis que serão lidas, precedidas pelo símbolo `&` (endereço), de acordo com a ordem estabelecida pela `"expressão de controle"`.

scan( ) - Exemplos:

Ler o valor de uma variável inteira:

```
scanf("%d", &j);
```

Ler o valor de duas variáveis inteiras:

```
scanf("%d %d", &i, &j);
```

Ler o valor de uma variável real e uma variável inteira:

```
scanf("%f %d", &x, &j);
```

Na função `scanf( )` é imprescindível o uso do símbolo de endereço `&` imediatamente antes do nome da variável.

Utilização

A função `scanf( )` não deve ser utilizada para exibir mensagens.

Isto não funciona!!!

```
scanf("Entre com o valor de i = %d", &i);
```

Isto sim, funciona!!!

```
printf("Entre com o valor de i = ");
scanf("%d", &i);
```

Formatando a saída

A função `printf()` permite definir como os valores das constantes e variáveis serão exibidos na tela.

Exibindo valores inteiros:

```
int i = 3;
printf("i = %d", i);          // i = 3
printf("i = %5d", i);         // i =     3
printf("i = %05d", i);        // i = 00003
```

Exibindo valores reais:

```
float pi = 3.14159265358;
printf("pi = %f", pi);        // pi = 3.14159265
printf("pi = %.4f", pi);      // pi = 3.1416
printf("pi = %8.2f", pi);     // pi =     3.14
```

Atenção!!!

Não se usa formatação de exibição na função `scanf()`.

Observação: operação de divisão

O símbolo / representa a operação de divisão. Uma expressão aritmética contendo diversos valores no numerador ou no denominador deve ser linearizada com o uso de parênteses.

$$x \leftarrow \frac{a+b}{c+d} \quad \rightarrow \quad \texttt{x = (a + b)/(c + d);}$$

O resultado da operação de divisão depende do tipo dos operandos na expressão.

Quando houver apenas operando inteiros...
Será realizada a divisão inteira.

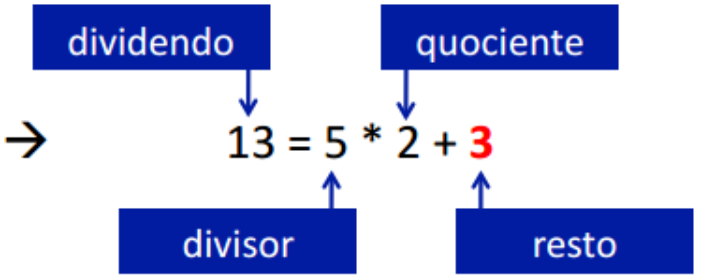
Quando pelo menos um dos operandos for real...
Será realizada a divisão real.

Observação: módulo

O símbolo % representa o operador módulo, que calcula o resto da divisão inteira entre 2 operandos do tipo inteiro.

Ex:

13	5
3	2




```
int D = 13, d = 5;
int Q, R;

Q = D/d;
R = D%d;

printf("Resultado da divisão inteira: %d\n", Q) ;
printf("Resto da divisão inteira: %d\n", R);
```

Exercício 1 – Ling. C

- Escreva programas em C que:
  - Leia dois números inteiros e exiba a soma, a diferença, a multiplicação, a divisão inteira, o resto e a potência entre eles;
  - Faça o mesmo do anterior, mas com 2 números reais (com a divisão de ponto flutuante e sem o resto);
  - Leia dois números, exiba-os e troque os valores das duas variáveis que receberam tais números, exibindo os mesmos depois da troca;
  - Faça o mesmo do anterior, mas usando apenas as 2 variáveis (sem usar variável auxiliar) para a troca dos valores.

FIM