

## Componentes e elementos da linguagem de programação C

## Sintaxe de linguagem de programação

A sintaxe consiste num conjunto de regras que definem a forma da linguagem, isto é, o formato como as sentenças podem ser escritas como sequências de componentes básicos, chamados palavras reservadas. Usando estas regras, pode-se identificar quando uma sentença está correta ou não. São regras que validam o formato das instruções que compõem um programa.

## Funções

Todo programa em C consiste em uma ou mais funções. A única função que necessariamente precisa estar presente é a denominada **main()**. Classificada como **função principal**.

## Estrutura básica de um programa C

Definição das bibliotecas (inicio do programa)

Definição de variáveis globais

main() (definição da função principal)

{

    declaração de variáveis locais

    instrução

.

.

} fim do programa

## Sintaxe de função

`tipo_de_retorno nome_da_função (lista_de_parametros)`

Onde:

`tipo_de_retorno` -> Indica o tipo de valor a ser retornado ao finalizar a execução função, sendo que, `void` indica que não existe retorno, e caso não seja informado o tipo de retorno o padrão será inteiro (`int`).

`nome_da_função` -> Indica o nome da função e o mesmo deve ser utilizado quando o programador identificar a necessidade que a mesma seja executada.

`lista_de_parametros` -> Os parâmetros possibilitam que se defina sobre quais dados a função deve operar.

## #include

A diretiva **#include** instrui o compilador a ler outro arquivo-fonte adicionado àquele que contém a diretiva **#include**.

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

```
void main()
{
    printf("Olaaaaaa!!!! Este é meu primeiro programa.");
}
```

# Algoritmos e Lógica a Programação

---

Fatec  
Santana de  
Parnaíba

CPS  
Centro  
Paula Souza

SÃO  
PAULO  
GOVERNO DO ESTADO

## Operadores Aritméticos em C

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

# Algoritmos e Lógica a Programação

---

Fatec  
Santana de Parnaíba

CPS  
Centro Paula Souza

SÃO PAULO  
GOVERNO DO ESTADO

## Operadores de incremento e decremento

C inclui dois operadores úteis geralmente não encontrados em outras linguagens. São os operadores de incremento e decremento, `++` e `--`. O operador `++` soma 1 ao seu operando, e `--` subtrai 1. Em outras palavras:

■ `x = x+1;`

é o mesmo que

■ `++x;`

e

■ `x = x-1;`

é o mesmo que

■ `x--;`

## Operadores Relacionais em C

### Operador

>  
>=  
<  
<=  
==  
!=

### Ação

Maior que  
Maior que ou igual  
Menor que  
Menor que ou igual  
Igual  
Diferente

## Operadores Lógicos

**Operador**

`&&`

`!!`

`!`

**Ação**

AND

OR

NOT

# Algoritmos e Lógica a Programação

Fatec  
Santana de Parnaíba

CPS  
Centro Paula Souza

SÃO PAULO  
GOVERNO DO ESTADO

## Operadores de atribuição

Linguagem  
C

=

Algoritmos

←

Exemplos com linguagem C:

soma = valor1 + valor2

valor = 5

Exemplos com algoritmo:

soma ← valor1 + valor2

valor ← 5

# Algoritmos e Lógica a Programação

Fatec  
Santana de Parnaíba

CPS  
Centro Paula Souza

SÃO PAULO  
GOVERNO DO ESTADO

## Operador sizeof

O operador **sizeof** é um operador em tempo de compilação unário que retorna o tamanho, em bytes, da variável ou especificador de tipo, em parênteses, que ele precede. Por exemplo, assumindo que inteiros são de 2 bytes e que **floats** são de 8 bytes,

```
float f;  
  
printf("%f", sizeof f);  
printf("%d", sizeof (int));
```

irá mostrar na tela 8 2.

## Variáveis e Constantes Tipos de dados da linguagem C

## Variáveis

Variáveis e constantes são os elementos básicos que um programa manipula dados. Uma variável é um espaço reservado na memória do computador para armazenar um tipo de dado determinado. Variáveis devem receber nomes para poderem ser referenciadas e modificadas quando necessário.

Ou seja, variáveis precisam de nomes(identificadores) para podermos acessa-las através do nosso programa.

## Tipos de dados primitivos (íntero)

Numérico íntero -> São valores inteiros, positivos, negativos ou zero. Como: variáveis que armazenam idade, quantidade de produtos, código de identificação, dentre inúmeros outros.

Este tipo de dado não aceita valores com casas decimais.

Exemplo de números inteiros: 1, 2, 3, 0, -1, 999,...

## Tipos de dados primitivos (íntero)

Numérico íntero -> São valores inteiros, positivos, negativos ou zero. Como: variáveis que armazenam idade, quantidade de produtos, código de identificação, dentre inúmeros outros.

Este tipo de dado não aceita valores com casas decimais.

Exemplo de números inteiros: 1, 2, 3, 0, -1, 999,...

## Tipos de dados primitivos (número Real ou ponto flutuante)

Valores com casas decimais. Como: variáveis que armazenam peso, altura, dinheiro, etc.

Exemplos: 1.6, 2.77, 3, 0.75, -1.87,...

## Tipos de dados primitivos (Caractere)

Tipo usado para armazenar letras. Como: variáveis que armazenam o gênero de uma pessoa, como F para feminino e M para masculino.

Exemplos: 'A', 'B', '1', "Carlos", "Minhas notas",...

## Tipos de dados primitivos (Booleano)

Variáveis desse tipo só podem armazenar um de dois valores: verdadeiro ou falso.

## Uso de variáveis em linguagem de programação

Para se usar uma variável em uma linguagem de programação é preciso criá-la. Para isso usa-se a seguinte sintaxe:

```
<tipo>  
<nome_da_variavel>;
```

Na linguagem de programação C, esse padrão é obrigatório e podemos usar os seguintes tipos primitivos: **int** (inteiro), **float** ou **double** (ponto flutuante), **char** (caractere) e **void** (sem valor). O tipo booleano é representado pelo comando **bool**, entretanto, para seu uso é necessário incluir a biblioteca `<stdbool.h>`.

## Regras para definição de nome de variáveis

- Todo nome só pode conter letras e dígitos;
- O caractere "\_" é contado como uma letra;
- Todo primeiro caractere deve ser sempre uma letra;
- Letras maiúsculas e minúsculas são consideradas caracteres diferentes em muitas linguagens, portanto é correto utilizar-se sempre as variáveis da mesma forma criando um padrão para evitar erros em outras linguagens;
- Palavras reservadas não podem ser usadas como nome de variáveis;
- É boa política escolher nomes que significam alguma coisa e indiquem a função da variável. Por exemplo: valor, soma, total, nome, raio.

## Tipos de variáveis e sua capacidade

A quantidade de espaço que será alocada depende do tipo de variável. O tamanho alocado na memória pelo tipo de variável limita o valor que pode ser guardado naquele espaço.

Tipo de dado	Significado	Tamanho (em bytes)	Intervalo de valores aceitos
char	Caractere	1	0 a 255
string	Cadeia de caractetes	*	*
byte	Inteiro de 8bits	1	-127 a 127
short	Inteiro curto	2	-32 768 a +32 767
int	Inteiro	4	-2 147 483 648 a 2 147 483 647
long	Inteiro longo	8	-922337203685477808 a 922337203685477807
float	Ponto Flutuante (real)	4	$3.4 \times 10^{-38}$ à $3.4 \times 10^{38}$
double	Ponto Flutuante duplo	8	$1.7 \times 10^{-308}$ a $1.7 \times 10^{308}$
boolean	booleano	1	true ou false

## Tipos de variáveis e sua capacidade

O programa abaixo desenvolvido utilizando a linguagem de programação C, irá disponibilizar no vídeo a quantidade de bytes em memória utilizada por cada tipo de dado.

```
1 # include <stdio.h>
2
3 main()
4 {
5     printf("float -> %d \r\n", sizeof(float));
6     printf("int -> %d \r\n", sizeof(int));
7     printf("double -> %d \r\n", sizeof(double));
8     printf("long -> %d \r\n", sizeof(long));
9     printf("char -> %d \r\n", sizeof(char));
10    printf("long int -> %d \r\n", sizeof(long int));
11    printf("short int -> %d \r\n", sizeof(short int));
12
13
14 }
15
```

## Constantes

O programa abaixo desenvolvido utilizando a linguagem de programação C, irá disponibilizar no vídeo a quantidade de bytes em memória utilizada por cada tipo de dado.

```
#define nome valor
```

A outra forma de se criar valores constantes é similar a criação de variáveis, porém antes do tipo usa-se o comando *const*, portanto a sintaxe ficará:

```
const tipo nome = valor;
```

## Exemplo de Constantes

```
1 #include<stdio.h>
2
3 #define pi 3.14
4
5 main()
6 {
7     const float g = 9.80;
8     printf("\n pi = %f",pi);
9     printf("\n g = %f",g);
10 }
```

## O endereço de memória de uma variável

As variáveis são usadas para reservar um espaço temporário na memória, que é dividida em blocos de *bytes*, e cada bloco possui um endereço que o identifica. Para sabermos o endereço de uma variável basta utilizarmos o operador & na hora de imprimir a variável.

```
1 #include <stdio.h>
2
3 main()
4 {
5     int x = 5;
6     int y = 10;
7     printf("\n Valor guardado em x: %d",x);
8     printf("\n Valor guardado em y: %d",y);
9     printf("\n Endereco de x: %x",&x);
10    printf("\n Endereco de y: %x",&y);
11 }
```

## Função printf()

É um comando de saída, o qual possui um vínculo com a biblioteca `stdio.h`. É utilizada quando se pretende obter uma resposta na tela do computador.

A sua síntese é definida por:

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

Exemplo:

```
printf ("O valor encontrado foi
%d", valor1);
```

## Formatações utilizadas na função printf()

Código	Função
%c	Permite a escrita de apenas um caractere.
%d	Permite a escrita de números inteiros decimais.
%e	Realiza-se a escrita de números em notação científica.
%f	É feita a escrita de números reais (ponto flutuante).
%g	Permite a escrita de %e ou %f no formato mais curto.
%o	Permite que números octais sejam escritos.
%s	Efetua-se a escrita de uma série de caracteres.
%u	Escreve-se um número decimal sem sinal.
%x	Permite a escrita de um número hexadecimal. [Sem título]
%n[]	Permite determinar entre colchetes quais caracteres podem ser ou não aceitos na entrada de uma sequência de caracteres, sendo "n" um valor opcional que determina o tamanho da sequência de caracteres.

# Algoritmos e Lógica a Programação

---

## scanf()

É um **comando de entrada**, isto é, são informações que possibilitam a entrada de dados pelo teclado, assim, a informação será armazenada em um determinado espaço da memória, como o nome e tipo específico da variável. A sintaxe é definida por uma expressão de controle (sempre entre aspas duplas) e pela lista de argumento.

Exemplo:

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

No exemplo, o computador entrará com um valor decimal e retornará o valor da variável “valor”.

O “&” é utilizado na função scanf() na lista de argumentos, sua função é retornar o conteúdo da variável, ou seja, retorna o endereço do operando.

A sintaxe da função scanf() é definida por:

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

# Algoritmos e Lógica a Programação

Fatec  
Santana de Parnaíba

CPS  
Centro Paula Souza

SÃO PAULO  
GOVERNO DO ESTADO

## scanf()

Código	Função
%c	Permite que seja efetuada a leitura de apenas um caractere.
%d	Permite fazer a leitura de números inteiros decimais.
%e	Permite a leitura de números em notação científica.
%f	É feita a leitura de números reais (ponto flutuante).
%l	Realiza-se a leitura de um número inteiro longo.
%o	Permite a leitura de números octais.
%s	Permite a leitura de uma série de caracteres.
%u	Efetua-se a leitura de um número decimal sem sinal.
%x	Permite que seja feita a leitura de um número hexadecimal.
%[código]	Permite que seja feita uma entrada formatada pelo código.

# Algoritmos e Lógica a Programação

---

Fatec  
Santana de Parnaíba

CPS  
Centro Paula Souza

SÃO PAULO  
GOVERNO DO ESTADO

## scanf()

```
#include <stdio.h>

int main()
{
    int valor;
    printf("Digite um número: ");
    scanf("%d",&valor);
    printf("\n o número é %d",valor);
    printf("\no endereço e %x",&valor);
    return 0;
}
```

# Algoritmos e Lógica a Programação

Fatec  
Santana de Parnaíba

CPS  
Centro Paula Souza

SÃO PAULO  
GOVERNO DO ESTADO

## scanf() --- Tabuada

```
#include <stdio.h>

int main()
{
    int valor, i;
    printf("Digite o numero da tabuada: ");
    scanf("%d",&valor);

    for (i=1; i <= 10; i++)
    {
        printf(" %d X %d = %d \r\n",i,valor, i * valor);
    }

    return 0;
}
```

## Instrução return

Para um programa retornar ao sistema operacional, é necessário utilizar a instrução retorna zero “return 0”.

O return também pode ser utilizado para retornar valor para a função que realizou a chamada de uma função auxiliar.

```
# include <stdio.h>
int main ( )
{
    int idade;
    printf ( "Digite a idade do candidato => " );
    scanf ( "%d" , &idade ) ;
    printf ( "O candidato tem %d anos !\n" , idade);
    return 0 ;
}
```

## Comentários e delimitadores

A medida que seus códigos na linguagem C forem aumentando, eles ficarão incrivelmente difíceis de serem entendidos por outra pessoa.

Sim, futuramente seu código será lido/alterado por outra pessoa, provavelmente você.

Para facilitar esse processo, você pode fazer 'comentários' em seus programas C, explicando o que cada trecho de código faz.

## Comentando códigos em C - Usando //

Sempre que quiser comentar alguma linha de seu código C, inicie a linha com duas barras: //

Todos os caracteres existentes após o “//” serão classificados como comentários até a finalização da linha <enter>.

```
#include <stdio.h>
int aux; // definicao de variavel
// Para Curso C Progressivo
int main()
{
    //O seguinte trecho mostra uma mensagem na tela
    //Essas linhas comentadas não irão aparecer na tela
    printf("Meu primeiro programa - C Progressivo!\n");
}
```

## Comentando códigos em C - Usando /\* \*/

Imagine agora que você precisa fazer um comentário de mais de 20 linhas. Isso é bem comum entre estudantes que estão resolvendo alguma questão e colam o enunciado e idéia da solução no corpo do código.

Todos os caracteres existentes entre o “/\*” e “\*/” serão classificados como comentários.

```
#include <stdio.h>
/* Adoro estudar a linguagem de programação C
enquanto escuto metal! Iron Maiden */

int main()
{
    printf("Meu primeiro programa - C Progressivo!\n");
}
```

## Questionário

1. Quais os operadores aritméticos em C?
2. O que é sintaxe de linguagem de programação?
3. Quais os operadores de incremento e decremento em C, e quais as instruções que cada um substitui?
4. Quais os operadores relacionais em C?
5. Quais os operadores lógicos em C?
6. Descreva a função printf().
7. Qual a sintaxe da função printf()?
8. O que a função scanf() possibilita?
9. Qual a sintaxe da função scanf()?
10. O que o operador sizeof() retorna?
11. Qual a funcionalidade da instrução “return”?
12. Para eu servem os comentários em um programa fonte?
13. Explique os dois tipos de marcação de comentários na linguagem de programação C.

## Questionário

14. Em linguagens de programação o que é uma variável?
15. Qual a diferença entre variáveis e constantes?
16. O que são valores inteiros e quais os tipos de dados que podem armazenar?
17. O que são valores real ou ponto flutuante e quais os tipos de dados que podem armazenar?
18. O que são valores caractere e quais os tipos de dados que podem armazenar?
19. O que são valores booleano e quais os tipos de dados que podem armazenar?
20. Na linguagem C qual sintaxe criar um variável?
21. Quais as regras (boas práticas de programação) para a definição de nome de variáveis?
22. Descreva a sintaxe para a criação de constantes na linguagem C.
23. Descreva o que é endereço de memória de uma variável e com qual operador em C que podemos acessar tal informação.