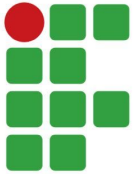


INSTITUTO FEDERAL
Bahia
Campus Santo Antônio de Jesus

http://www.portal.ifba.edu.br/santoantonio

LINGUAGEM DE PROGRAMAÇÃO

Prof. George Pacheco Pinto

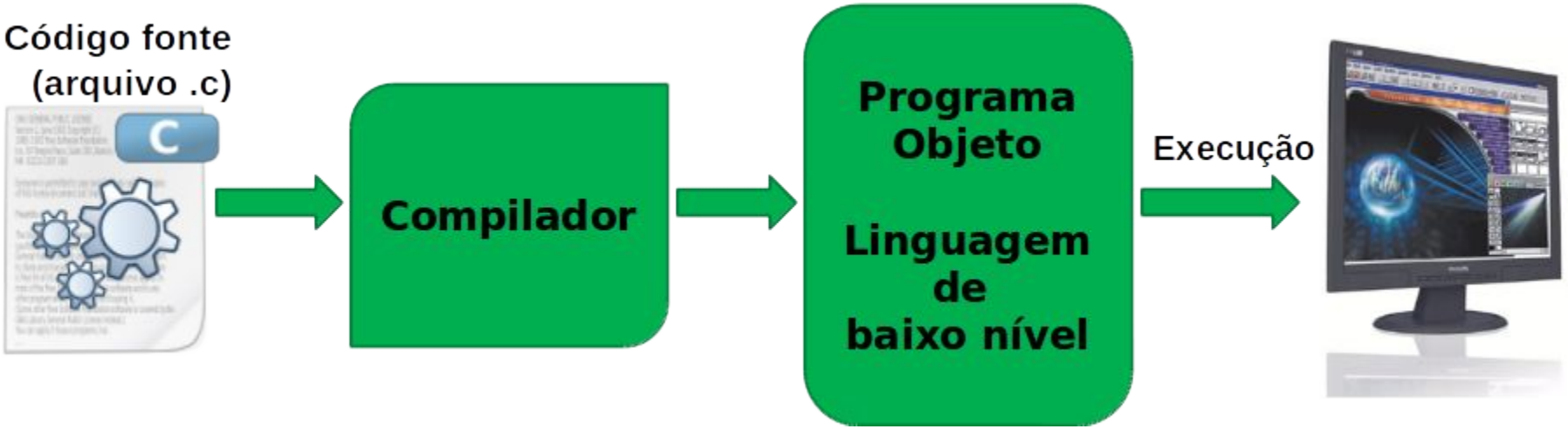
Agenda

- Linguagem C
  - Características
  - Estrutura básica de um Programa em C
    - Tipos primitivos
    - Declaração de Variáveis
    - Entrada de dados
    - Saída de dados
    - Operadores:
    - Aritméticos
    - Relacionais
    - Lógicos
    - Precedência de Operadores
    - Expressões

HISTÓRICO

- Necessidade de se escrever programas que utilizassem recursos próprios da linguagem de máquina de uma forma mais simples e portável que o assembly;
- Desenvolvida nos laboratórios Bell na década de 70 por Brian Kernighan e Dennis M. Ritchie;
- Padronização de 82 a 89 (C ANSI);
- Versão atual C18 - 2018.

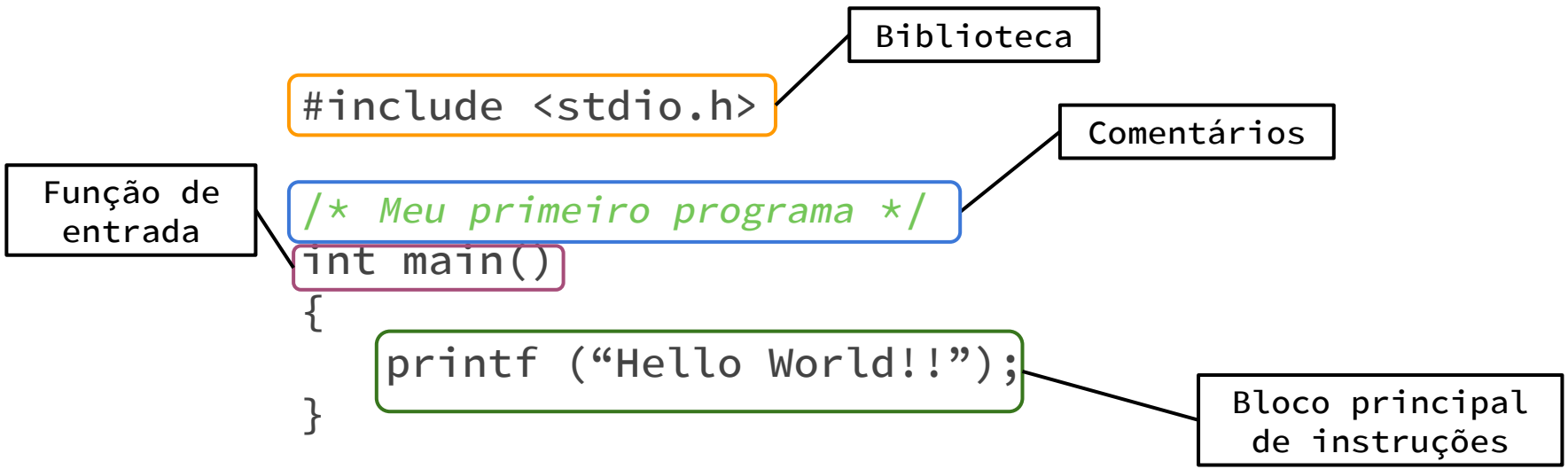
Linguagem compilada



CARACTERÍSTICAS

- Programas em C são compilados, gerando programas executáveis;
- Case-sensitive, ou seja, faz distinção entre maiúsculo e minúsculo;
- É uma linguagem estruturada. Permite a divisão do programa em módulos;
- Tem como principal componente estrutural a função, que corresponde a um bloco de construção em que toda atividade do programa ocorre.

ESTRUTURA BÁSICA



BIBLIOTECAS

- Funções são uma forma genérica de resolvermos problemas;
- Em geral, utilizamos algumas funções já prontas para fazer determinadas tarefas que são consideradas básicas;
- Estas funções encontram-se geralmente em Bibliotecas;
- **#include** informa ao compilador para incluir na compilação do programa outros arquivos, normalmente bibliotecas de funções ou rotinas de usuários.

MAIN

- Todos os programas em C tem essa função, que será chamada quando o programa for executado;

- Sem main o programa em C não compila;

- Um programa começa executando a função main(). E termina, quando esta função termina.

```
...
main {
    ....
}
```

Delimitadores

Blocos - { }

Comandos - ;

Comentários

```
/*
  Bloco de Comentário
*/

// Comentário de Linha
```

Palavras reservadas

Toda linguagem de programação têm palavras reservadas, ou seja, não podem ser usadas a não ser nos seus propósitos originais.

ANSI C			
break	double	int	struct
case	else	long	switch
char	enum	return	unsigned
const	float	short	void
continue	for	signed	while
default	goto	sizeof	static
do	if		

VARIÁVEIS

- Identificação textual de um espaço na memória, que objetiva armazenar uma informação de forma temporária.

X	10
Y	5
VALOR	15.25
NOME	GEORGE
CONT	10
TOTAL	100

Tipos de dados

- **char** - O valor armazenado é um caractere;
- **int** - Número inteiro;
- **float** - Número em ponto flutuante de precisão simples. São conhecidos normalmente como números reais;
- **double** - Número em ponto flutuante de precisão dupla;
- **void** - Este tipo serve para indicar que um resultado não tem um tipo definido. Uma das aplicações deste tipo em C é criar um tipo vazio que pode posteriormente ser modificado para um dos tipos anteriores

TIPOS DE DADOS

Modificadores

- signed: indica número com sinal (inteiros e caracteres)
- unsigned: número apenas positivo (inteiros e caracteres)
- long: aumenta abrangência (inteiros e reais)
- short: reduz a abrangência (inteiros)

TIPOS DE DADOS

Tipo	Nº de bits	Formato para leitura com scanf
char	8	%c
unsigned char	8	%c
signed char	8	%c
int	16	%i
unsigned int	16	%u
signed int	16	%i
short int	16	%hi
unsigned short int	16	%hu
signed short int	16	%hi
long int	32	%li
signed long int	32	%li
unsigned long int	32	%lu
float	32	%f
double	64	%lf
long double	80	%Lf

https://en.wikipedia.org/wiki/C_data_types#cite_note-c99sizes-3

CONSTANTES

- São valores que são mantidos fixos pelo compilador;

- Também podem ser:
  - Octais - 0NUMERO_OCTAL
  - Hexadecimais - 0xNUMERO_HEXADECIMAL

- Exemplos:
  - ‘\n’ (caractere), “C++” (string), 10 (inteiro), 15.0 (float), 0xEF (239 em decimal), 03212 (1674 em decimal)

CONSTANTES

Código	Significado
\b	Retrocesso (backspace)
\f	Alimentação de Formulário (form feed)
\t	Tabulação Horizontal (tab)
\n	Nova Linha
\”	Aspas
\’	Apóstrofo
\0	Nulo
\\	Barra Invertida

DECLARAÇÃO DE CONSTANTE

- Usando **#define**

```
#include <stdio.h>
#define PI 3.14159

int main (){
  double r=5.0;
  double circle;
  circle = 2 * PI * r;
 ...
}
```

- Usando **const**

```
const int tamanho = 100;
const char tabul = '\t';
const int codigo = 12440;
```

Nomeando Variáveis

- Começar com letra ou _
- Conter letras, números ou _
- Não podem ter o mesmo nome que as palavras reservadas
- Não podem conter espaços em branco
- Ter até 31 caracteres
- Não utilize acentos
- C é CaseSensitive
- Ex:engenharia, Engenharia, enGenharia

DECLARAÇÃO DE VARIÁVEIS

- podemos declarar variáveis:
  - fora de todas as funções do programa (globais);
  - no início de um bloco de código (locais);
  - na lista de parâmetros de uma função;

```
int main()   {
    char condicao;
    int i;
    for (i=0; ...)  {
            /* Bloco do for */
            float f2;
            ...
        func1(i);
    }
        ...
    return(0);
}
```

INICIALIZANDO / ATRIBUINDO VALORES À VARIÁVEIS

- inicializando variáveis:
  - inicializando ao declará-las

```
int contador=1;
```

  - inicializando depois de declará-las

```
int contador;
contador=1
```

ENTRADA DE DADOS

- Aguarda que o usuário digite um valor e atribui o valor informado em uma dada variável.

  scanf (string-de-controle,lista-de-argumentos);

- Parâmetros:
  - uma string, indicando os tipos das variáveis que serão lidas
  - uma lista de variáveis.

Algumas formatações para leitura

```
%c - leitura de caracter
%d - leitura de número inteiro
%f - leitura de número real
%s - leitura de caracteres
```

ENTRADA DE DADOS

Exemplo

```
int variavel;
float variavel2
scanf (“%d %f”, &variavel, &variavel2);
```

IMPORTANTE!!!

SEMPRE COLOCAR O

&

ANTES DAS VARIÁVEIS
NO SCANF

IMPORTANTE 2

Antes de ler caracteres é necessário limpar a entrada padrão. Isso é muito importante caso antes tenha sido utilizado o scanf.

fflush(stdin);

Exemplo

```
...
char nome[20];
char sexo;
scanf("%s", &nome);
fflush(stdin);
scanf("%c", &sexo);
```

Função Saída (printf)

- Imprime um conjunto de textos e valores na tela.

  printf(“<string de saída>”,<variáveis>);

- A string de saída pode ser composta por TEXTO, STRING DE CONTROLE e CARACTERES DE COMANDO.

Algumas formatações para escrita

```
%c - escrita de caracter
%d - escrita de número inteiro
%f - escrita de número real
%s - escrita de caracteres
```

Função Saída (printf)

- `printf (“O nome digitado foi %s\n”, nome);`
- `printf (“Sexo: %c\n”, sexo);`
- `printf (“Salário: %.2f\n”, salario);`
- `printf (“Matrícula: %d\n”, matricula);`

Saída de Dados

- Caracteres de comando
  - \n - pula uma linha
    ```
    printf(“escreve e pula linha \n”);
    ```
  - \t - tabulação horizontal
    ```
    printf(“escreveu e \t tabulou”);
    ```
  - \” - escreve aspas
    ```
    printf(“vai colocar entre \”aspas\” “);
    ```
  - \a beep sonoro
    ```
    printf(“vai mostrar e dar um beep no final \a”);
    ```

Exibindo acentos

```
#include <locale.h>

setlocale(LC_ALL, "Portuguese");
```

EXERCÍCIOS

1. Escreva um programa que declare 3 variáveis, atribua a elas os valores 5, 6 e 7 e depois apresente a seguinte mensagem: “O valor da segunda variável é....”

2. Faça um programa para ler os dados de um aluno do curso de SI: matrícula, idade, salário e sexo (M ou F). Após a leitura deve-se apresentar os dados digitados na forma de um relatório.

OPERADORES ARITMÉTICOS

Operador	Descrição
+	Operador de adição
-	Operador de subtração
*	Operador de multiplicação
/	Operador de divisão
%	Operador de resto de uma divisão de números inteiros
++	Operador de incremento
--	Operador de decremento

OPERADORES RELACIONAIS

Operador	Descrição
>	Maior que
<	Menor que
>=	Maior que ou igual a
<=	Menor que ou igual a
==	Igual a
!=	Diferente

OPERADORES LÓGICOS

Operador	Descrição
&&	AND lógico
||	OR lógico
!	NOT lógico

OPERADORES COMBINADOS

- O operador de atribuição pode ser combinado com outros operadores aritméticos.

Exemplos:

```
a += b; equivalente a a = a + b;
a -= 2; equivalente a a = a - 2;
a *= 1+1; equivalente a a = a * (1+1);
a %= b*c+d; equivalente a a = a % (b*c+d);
```

Incremento Pós e Pré-Fixado

Pós-Fixado

```
x= 2;

y = x++; // x será 3; y será 2
```

1º y = x -> y=2
2º x+1 -> x=3

Pré-Fixado

```
x = 2;

y = ++x; // x será 3; y será 3
```

1º x+1 -> x=3
2º y=x -> y=3

Decremento Pós e Pré-Fixado

Pós-Fixado

```
x= 2

y = x--; // x será 1; y será 2
```

1º y = x -> y=2
2º x-1 -> x=1

Pré-Fixado

```
x = 2;

y = --x; // x será 1; y será 1
```

1º x-1 -> x=1
2º y=x -> y=1

Precedência de Operadores

- Qual o valor da expressão 5 + 10 % 3?
- E da expressão 5 * 10 % 3?

- Precedência é a ordem na qual os operadores serão calculados quando o programa for executado. Em C , os operadores são calculados na seguinte ordem:
  - * e /, na ordem em que aparecem na expressão.
  - %
  - + e -, na ordem que aparece na expressão.

Precedência de Operadores

a)5 + 10 % 3?

5 + 1

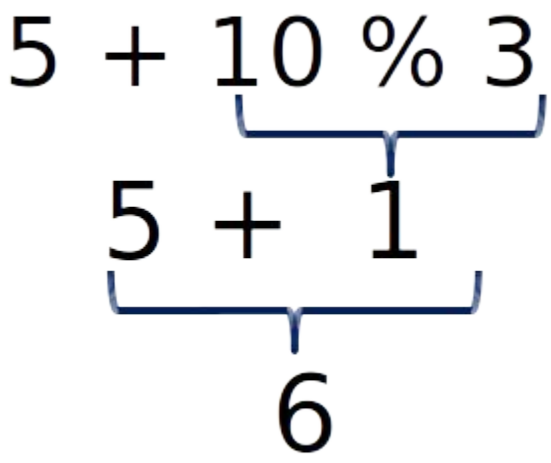
6

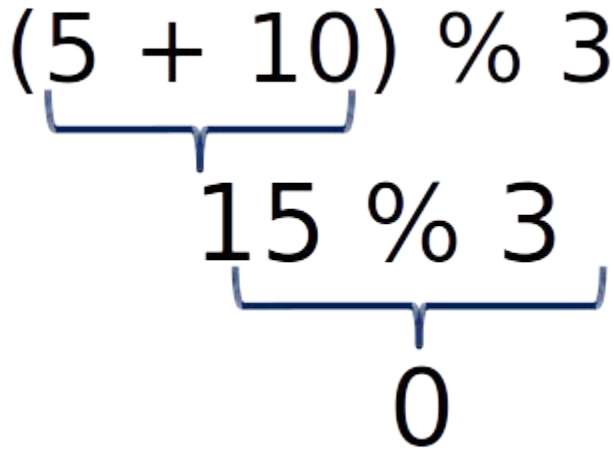
b)5 * 10 % 3?

50 % 3

2

ALTERANDO A PRECEDÊNCIA





CONVERSÃO DE TIPOS (CAST)

- Os tipos de variáveis e os valores que lhes são atribuídos têm de coincidir:
  `int teste = “vinte”;  // ERRO NUNCA FAZER`

- Para efetuarmos uma conversão de um tipo maior para outro de menor capacidade é necessário realizar conversão.

CONVERSÃO DE TIPOS (CAST)

- **Exemplo 1**

```
float a = 5.25;
int b = (int)a;
```

- **Exemplo 2**

```
char c = ’A’;
int x = (int)c;
```

CONVERSÃO DE TIPOS (CAST)

- **Exemplo 1**

```
float a = 5.25;
int b = (int)a;    /*Casting explícito float para int.  b = 5 */
```

- **Exemplo 2**

```
char c = 'A';
int x = (int)c;    /*Casting explícito char para int.  x = 65*/
```

CONVERSÃO DE TIPOS (CAST)

- **Exemplo 3**

```
int x=7, y=5 ;
float z;
z=x/y;
```

- Se quisermos pegar o valor exato de 7/5 então é preciso realizar a conversão explícita int para float:

```
int x=7, y=5;
float z;
z = (float)x/(float)y;
```

CONVERSÃO DE TIPOS (CAST)

- **Exemplo 3**

```
int x=7, y=5 ;
float z;
z=x/y;  /*Aqui z = 1*/
```

- Se quisermos pegar o valor exato de 7/5 então é preciso realizar a conversão explícita int para float:

```
int x=7, y=5;
float z;
z = (float)x/(float)y;  /*Aqui z =1.4*/
```

EXPRESSÕES

- Expressões compostas
(<expressão>/<expressão>)+<expressão>
((<expressão>..+.-./.%.<expressão>))

- Você pode usar quantos parênteses desejar dentro de uma expressão, contanto que utilize o mesmo número de parênteses para abrir e fechar expressões.

EXPRESSÕES

```
i == 3;  // igual
i != 3;   // diferente
i > 3; // maior
i < 3; // menor
i <= 3;  // menor ou igual
i >= 3;  // maior ou igual

Operações lógicas

b = b1 && b2; // AND
b = b1 || b2; // OR
b = !b1; // NOT
```

EXERCÍCIOS

3) Escreva um programa em C que leia 3 números inteiros, calcule a média aritmética e escreva a resposta para o usuário.

4) Faça um programa em C que, dado o valor de uma compra, calcule o ICMS a ser pago. Considere uma taxa de 12%.

5) Dado um valor em anos, faça um programa em C que converta para segundos.

EXERCÍCIOS

6) Elabore um programa em C para ler um valor em reais e convertê-lo para dólar.

7) Elabore um programa em C para ler a descrição, quantidade e valor unitário de 2 produtos. Estes produtos compõem o estoque de uma empresa. Calcular e imprimir o valor total do estoque da empresa.

8) Elabore programa em C que leia um número com 3 dígitos e o imprima invertido.

Referências

Consultar ementário.