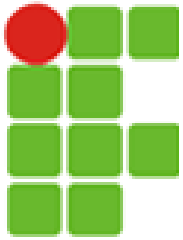


INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO

Linguagem de Programação Estruturada – LPES1

Tecnologia em Análise e Desenvolvimento de Sistemas

Introdução à Programação

Fernando Vieira Duarte
fernandoduarte@ifsp.edu.br

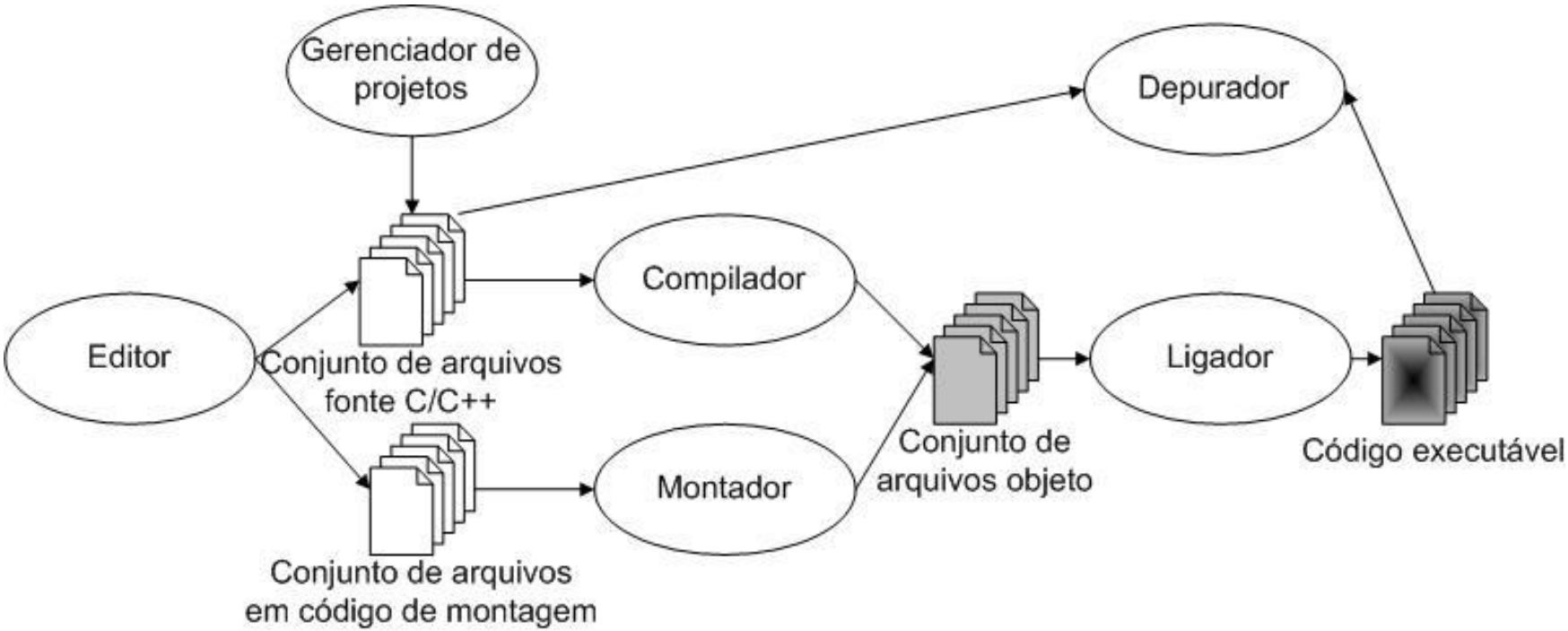
Aula 3 - 17/02/2020

Linguagem de Programação

LP (**Linguagem de Programação**) é uma maneira distinta de dar instruções ao computador sobre **o que e como fazer**, ou seja, é uma linguagem:

- destinada ao uso de uma pessoa **[1]**
- para expressar um processo **[2]**
- por meio do qual um computador **[3]**
- pode resolver um problema **[4]**.

Processo de Compilação e Geração de Código Executável



Sintaxe e semântica de uma linguagem de programação

- FORMA → SINTAXE
- SIGNIFICADO → SEMÂNTICA

Erros sintáticos: < int nome1**;** nome2**;** > em vez de
< int nome1**,** nome2**;** >

Erros semânticos: int termo; termo = 5**,**8

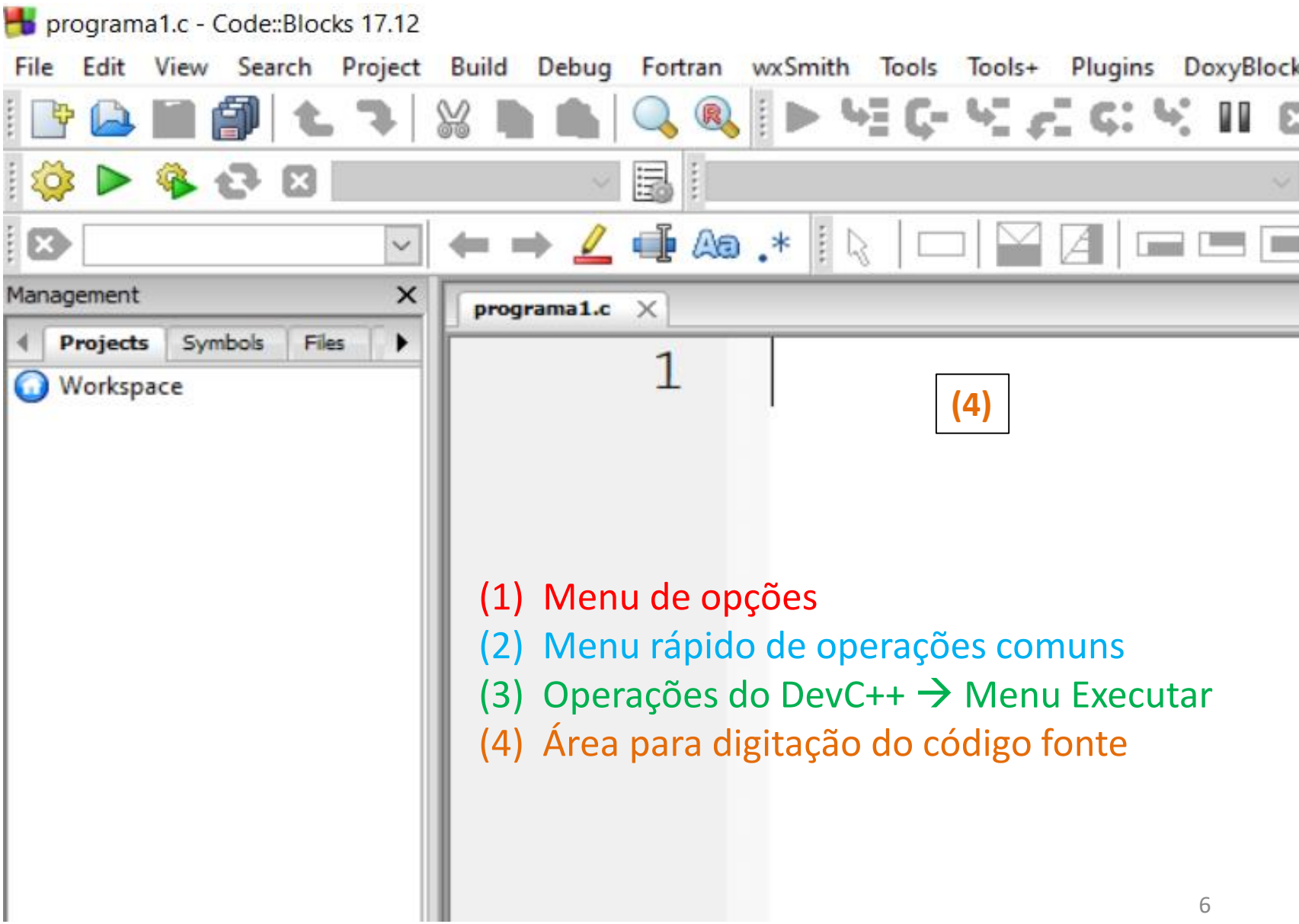
Erros léxicos: uso indevido de caracteres ou termos que não são permitidos pela linguagem de programação. Como exemplo, o uso do caractere @ em um programa C.

Ambientes de Desenvolvimento C

- ***Netbeans*** - gratuito e de código-fonte aberto, distribuído pela *Sun Microsystems. É* multiplataforma, ou seja, há distribuições para diversos sistemas operacionais, incluindo *Windows* e *Linux*. *Netbeans* também suporta diversas linguagens de programação, incluindo Java e C. Disponível em: <http://www.netbeans.org/>.
- ***Code::Blocks*** – ou apenas C::B, é uma IDE para C e C++. Esta IDE também é gratuita, de código aberto e multiplataforma, com distribuições para *Windows* e *Linux*. Disponível em: <http://www.codeblocks.org>.
- ***Microsoft Visual C++ Express Edition*** – disponível pela *Microsoft* para desenvolvimento de códigos em C e C++. A expressão "*Express Edition*" é usada para classificar um conjunto de ambientes de desenvolvimento que a Microsoft disponibiliza de forma gratuita. Disponível em: <http://www.microsoft.com/Express/vc/>.
- ***Eclipse for C/C++ Developers*** – Disponível para *download* em: <http://www.eclipse.org>.
- *Borland C++* - disponível em: <http://www.codegear.com/downloads/free/cppbuilder>.
- ***Dev-C++*** - disponível: <http://www.bloodshed.net/devcpp.html>.

IDE (*Integrated Development Environment*)

Code::Blocks



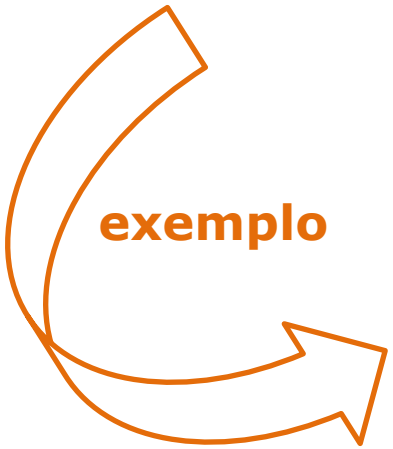
(1) Menu de opções
(2) Menu rápido de operações comuns
(3) Operações do DevC++ → Menu Executar
(4) Área para digitação do código fonte

Linguagem C

- *Case Sensitive*.
- Toda instrução (ou comando) termina com o caractere ponto e vírgula (**;**).
- Números reais separados por ponto (**.**). Ex.: 25.98
- Necessita ser compilado e executado – processos diferentes.

Estrutura básica de um programa escrito em linguagem C

```
1.  #include <stdio.h>
2.  #include <stdlib.h>
3.  int main()
4.  {
5.      // aqui vem todos
6.      // os comandos
7.      return 0;
8.  }
```




```
1.  #include <stdio.h>
2.  #include <stdlib.h>
3.  int main()
4.  {
5.      printf("Texto que sera exibido na tela!");
6.      system("PAUSE");
7.      return 0;
8.  }
```

Estrutura básica de um programa escrito em linguagem C

1. #include <stdio.h>

Standard In/Out, contém rotinas necessárias para a entrada e saída padrão de dados.

Bibliotecas incluem funções pré-escritas por outros programadores que auxiliam no desenvolvimento do código, sem a necessidade de reescrever algumas rotinas.

Exemplos de funções da stdio: fprintf, printf, fscanf, scanf, getchar, putc, getc, fopen, fclose.

#include permite a inclusão de um arquivo de cabeçalho.

2. #include <stdlib.h>

Standard Library, contém várias funções de uso geral, incluindo gerenciamento dinâmico de memória, geração de números aleatórios, comunicação com o ambiente, aritmética de números inteiros, pesquisa, classificação e conversão.
Exemplos de funções da stdio: system, exit, rand, srand, malloc, free.

Estrutura básica de um programa escrito em linguagem C

1. int main**()**
2. **{**
3. *//aqui vem todo*
4. *//o corpo do programa*
5. **}**

Todo programa C deve, obrigatoriamente, ter esta função.
Main determina o local onde os comandos do programa são escritos e delimita seu bloco de instruções por meio de um par de chaves

Estrutura básica de um programa escrito em linguagem C

```
1. #include <stdio.h>
2. #include <stdlib.h>

3. int main()

4. {

5. printf("Texto que sera exibido na tela!");

6. system("PAUSE");

7. return 0;

8. }
```

exibe informações (que estiverem dentro dos parênteses e aspas) na tela

é útil para causar uma pausa após a execução do código e manter a janela aberta

serve para informar ao compilador que ocorreu tudo certo com a função main(). Se main retornasse um outro valor diferente de 0 haveria um problema em sua execução, que seria informada ao compilador.

Variáveis

Variável é uma posição, frequentemente localizada na memória do computador, que armazena um único valor de cada vez.

As variáveis existem somente durante a execução de um programa (em tempo de execução), perdendo seu conteúdo quando o programa se encerra.

Cada variável é identificada na memória por meio de um endereço único, em formato hexadecimal, conhecido pelo sistema operacional.

```
tipo_dado  nome_variavel;
```

Variáveis e Tipos de Dados 1/2

Os cinco tipos de dados da linguagem C são os seguintes:

- ***char***: é utilizado para armazenar valores do **tipo caractere**, representados por aspas simples, por exemplo 'A', 'a', 'B', 'b', '@', '$'.
  - ❑ Ocupam 8 *bits* (1 *byte*) na memória.

- ***int***: armazena dados do **tipo inteiro** e é o mais utilizado na linguagem C.
  - ❑ O tamanho do conjunto que pode ser representado normalmente depende da máquina em que o programa está rodando, mas possui tamanho aproximado de 16 *bits* (2 *bytes*).

- ***float***: utilizado para armazenar número em **ponto flutuante** de precisão simples. São conhecidos normalmente como números reais, exemplos: 5,52; .9,25; 47,38.
  - ❑ Um dado do tipo *float* ocupa 32 *bits* (4 *bytes*) na memória principal e possui 6 dígitos de precisão (0.123456).

Variáveis e Tipos de Dados 2/2

•***double***: também utilizado para armazenar número em **ponto flutuante**, porém com precisão dupla – de até 10 dígitos (0.1234567890).

- ❑ Um dado do tipo *double* ocupa 64 *bits* (8 *bytes*) na memória.

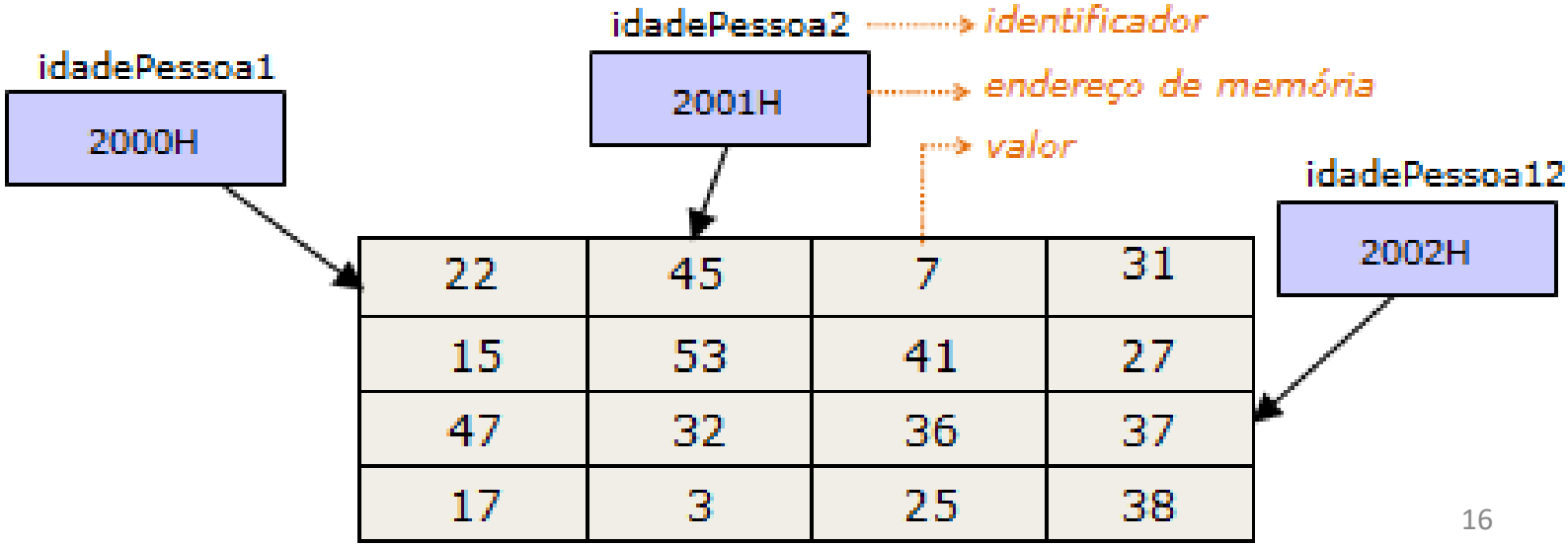
•***void***: este tipo serve para indicar **ausência de valor**. Uma das aplicações deste tipo em C é criar um tipo vazio que pode posteriormente ser modificado para um dos tipos anteriores. Todavia, não é comum associar o tipo *void* a variáveis dentro do programa; ele é usualmente utilizado na especificação de retorno de valores de funções ou criação de ponteiros.

Exemplo 1

```
1.  //Duas barras indicam comentários de uma linha no código.
2.  /* Barra seguida de asterisco indica comentário de várias linhas.
3.     O mesmo será finalizado quando encontrar: asterisco seguido por barra */
4.  //Comentários serão ignorados pelo compilador durante a execução.
5.  //Você pode fazer comentários que ajudem a compreender cada trecho do seu código.
6.  //É usual indicar uma descrição sobre o que o programa faz
7.  // e também o nome do programador logo no início do código, da seguinte forma:
8.  //Este código consiste de um exemplo de estrutura da linguagem C.
9.  //Autor: Professor
10. //cabeçalho do programa com chamada à biblioteca
11. #include <stdio.h>
12. #include <stdlib.h>
13. int main()
14. {
16.    //declaração de variáveis
17.    int idade;
18.
19.    //atribui o valor 20 à variável idade. O sinal = indica atribuição.
20.    idade = 20;
21.
22.    //exibe o valor de idade na tela
23.    printf("Idade = %i \n", idade);
24.
25.    system("PAUSE");
26.    return 0;
27. }
```

Exemplo 2 – o que acontece na memória?

```
1.  #include <stdio.h>
2.  #include <stdlib.h>
3.  int main()
4.  {
    4.  //declaração de variáveis
    5.  int idade;
    6.
    7.  //atribui o valor 20 à variável idade. O sinal = indica atribuição.
    8.  idade = 20;
    9.
    10. //exibe o valor de idade na tela
    11. printf("Idade = %i \n", idade);
    12. system("PAUSE");
    13. return 0;
14. }
```



Funções printf() e scanf() – Exemplo 3

```
1.  //Utilizando os comandos scanf e formatação de saída de dados.
2.  //Autor: Professor
3.  #include <stdio.h>
4.  #include <stdlib.h>
5.  int main()
6.  {
7.      //declaração de variáveis
8.      int idade, anoNasc, qtdeAnos;
9.      int anoAtual = 2012; //atribuição de valor durante a declaração da variável
10.
11.     printf("Digite sua idade: ");
12.     scanf("%d", &idade); //atribuição valor a partir do teclado
13.
14.     //exibe o valor de idade na tela
15.     printf("Idade informada pelo usuario = %i \n \n", idade);
16.
17.     printf("Qual ano voce nasceu? >> ");
18.     scanf("%d", &anoNasc);
19.
20.     qtdeAnos = anoAtual – anoNasc; //atribuição valor a partir de cálculo
21.
22.     //imprime valores de variáveis na tela
23.     printf("\nIdade informada pelo usuario = %i \n", idade);
24.     printf("Idade que tera ao final deste ano (calculada a partir do ano de
        Nascimento): %i \n\n", qtdeAnos);
25.
26.     system("PAUSE");
27.     return 0;
28. }
```

Funções printf() e scanf()

A função **printf( )** possui **k parâmetros adicionais**, onde k corresponde ao número de especificadores de formato que ocorrem no primeiro parâmetro. Os especificadores de formato mais comuns são:

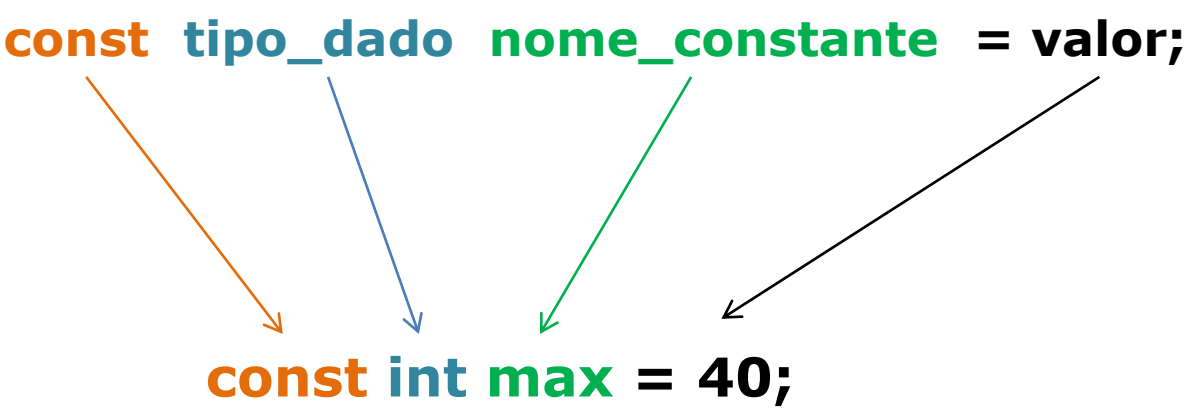
- **%c** para caractere (e.g., tipo de dados **char**),
- **%d** ou **%i** para números inteiros (e.g., tipo de dados **int**),
- **%f** para números reais exibidos com parte inteira, ponto e parte fracionária (e.g., tipos de dados **float** e **double**),
- **%lf** para números reais exibidos com parte inteira, ponto e parte fracionária (**double**),
- **%e** para números reais exibidos em notação científica (e.g., tipos de dados **float** e **double**, tal como 1.034e+017),
- **%u** para **endereços de memória**, e
- **%s** para cadeias de caracteres chamadas.

Exemplo 4

```
1.  //Mais sobre o comando scanf.
2.  //Autor: Professor
3.  #include <stdio.h>
4.  #include <stdlib.h>
5.  int main()
6.  {
7.
8.      //declaração de variáveis
9.      int idade;
10.     float peso;
11.     char nome[30];
12.
13.     printf("Digite seu nome, idade e peso (nesta ordem e separados por <ENTER> ): \n");
14.     scanf("%s%d%f", &nome, &idade, &peso);
15.
16.     //exibe o valor de idade na tela
17.     printf("\n\nDADOS INDORMADOS: \n");
18.     printf("_____\n");
19.     printf("Nome: %s, \nIdade: %i, \nPeso: %f \n \n", nome, idade, peso);
20.
21.     system("PAUSE");
22.     return 0;
23. }
```

Constantes

Constantes são **valores** armazenados em memória que possuem o **mesmo valor do início ao término** do programa. Podem ser comparadas às variáveis em termos de espaço reservado na memória, no entanto não podem ter seu valor alterado durante a execução.



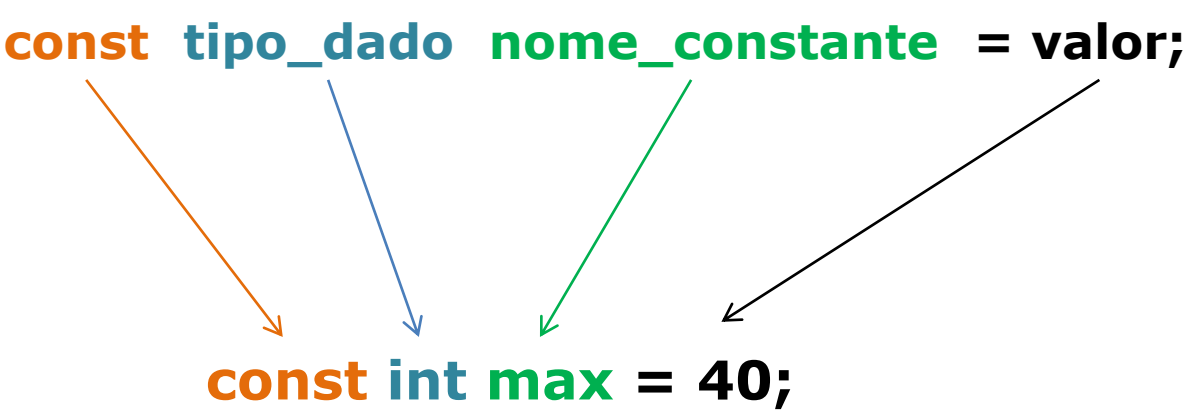
Exemplo 5 - constantes

```
1.  //Exemplo de uso de constante.
2.  //Autor: Professor
3.
4.  #include <stdio.h>
5.  #include <stdlib.h>
6.
7.  int main()
8.  {
9.      //declaração de variáveis e constantes
10.     const int max = 40;
11.     char nome[max];
12.
13.     printf("Digite seu nome: ");
14.     gets(nome);

15.     printf("\nNome: %s \n", &nome);
16.
17.     system("PAUSE");
18.     return 0;
19. }
```

Também podemos declarar constantes usando o comando *define*

O que é diferente do ***#define***, que na verdade não faz parte da linguagem C, é algo que vem antes, é um comando pro pré-processador que ocorre antes da compilação.



Exemplo 5 - constantes

```
1.  //Exemplo de uso de constante.
2.  //Autor: Professor
3.
4.  #include <stdio.h>
5.  #include <stdlib.h>
6.
7.  int main()
8.  {
9.      //declaração de variáveis e constantes
10.     const int max = 40;
11.     char nome[max];
12.
13.     printf("Digite seu nome: ");
14.     gets(nome);

15.     printf("\nNome: %s \n", &nome);
16.
17.     system("PAUSE");
18.     return 0;
19. }
```

Identificadores

Há específicas para criação de identificadores na linguagem C e são as seguintes:

1. não podem conter caracteres especiais (com exceção do *underline* e cifrão, que são aceitos), tais como -, (,), /, &, %, espaço em brando, etc;
2. não pode iniciar com número;
3. não pode ser uma palavra reservada da linguagem C (possui 32 palavras reservadas no total), tais como: **auto, break, case, char, const, continue, default, do, double, else, enum, extern, float, for, goto, if, int, long, register, return, short, signed, sizeof, static, struct, switch, typedef, union, unsigned, void, volatile e while**.

- **Exemplos de identificadores válidos e formas de utilização**:

```
 int ano; x = 246;
 int numero = 27;
A32 = 43;
usuario_002;
$ano = 2012.
```

A linguagem C é *case-sensitive*.

Funções getchar(), getch() e gets()

getchar() → lê o caractere digitado pelo usuário e aguarda que a tecla <ENTER> seja pressionada;

getch() → lê o caractere digitado pelo usuário , a leitura do teclado é automática;

gets() → lê um conjunto de caracteres separados por espaço, por exemplo: "Gislaine Cristina Micheloti Rosales". Funciona somente para o tipo char.

Leitura a partir do teclado: Exemplo 6

```
1.  #include <stdio.h>
2.  #include <stdlib.h>
3.  int main()
4.  {
5.      //declaração de variáveis e constantes
6.      char sexo;
7.      char letra;
8.      char nome[40];
9.      char telefone[13];
10.
11.     printf("Digite seu nome completo: ");
12.     gets(nome);
13.
14.     printf("Digite seu telefone: ");
15.     gets(telefone);
16.
17.     printf("Digite seu sexo [F ou M]: ");
18.     sexo = getchar();
19.
20.     printf("Digite uma letra qualquer do alfabeto [a..Z]: ");
21.     letra = getch();
22.
23.     printf("\n\nNome digitado: %s \n", &nome);
24.     printf("\nTelefone digitado: %s \n", &telefone);
25.     printf("\nSexo digitado: %c \n", sexo); //observe que não há o caractere & na
        impressão
26.     printf("\nLetra digitada: %c \n", letra);
27.     system("PAUSE");
28.     return 0;
29. }
```

Dúvidas

Exercícios

1. Elaborar um programa que converte temperaturas em Fahrenheit para graus Célsius. Utilize a seguinte fórmula: F = 180(C + 32)/100.

2. Escreva um programa na Linguagem C que leia 3 valores: a, b, c e calcule e escreva a média aritmética.

Calcular a média aritmética pela fórmula: $\frac{a + b + c}{3}$

3. Desenvolver um programa que calcule a seguinte expressão:

$$\left(\frac{26}{12}\right) + 5 * \left(\frac{(10 + (20/3))}{(9\%2)}\right) - (4 * (3 - 8))$$

4. Construa um programa que calcule a área de um triângulo. Lembre-se de que para realizar esse cálculo é necessário utilizar a fórmula:
área = (base * altura)/2

5. Faça um programa na Linguagem C que receba o salário-base de um funcionário, calcule e mostre o salário a receber, sabendo-se que esse funcionário tem gratificação de 5% sobre o salário-base e paga imposto de 7% sobre o salário-base.

Exercícios

6. Escreva um programa em C que solicite ao usuário a altura e o raio de um cilindro, e imprima o volume do cilindro. O volume do cilindro é calculado pela seguinte fórmula: volume = 3.141592 * raio * raio * altura.

7. Você foi contratado por uma empresa de construção para fazer um programa em C que calcule o salário líquido dos operários no fim de cada mês, sabe-se que cada operário recebe R$ 10,00 por cada hora trabalhada, e que se desconta 8% do salário bruto para o INSS. Desta forma, o usuário deverá informar o número de horas trabalhadas de um operário e o algoritmo retornará o salário líquido do operário.

8. Você foi contratado por uma loja de eletrônicos para fazer um programa em C que calcule a conversão de dólares para real, sabe-se que a aplicação deverá ler a cotação do dólar do dia e o valor a ser convertido.

9. Faça programa em C que receba o ano de nascimento de uma pessoa e o ano atual, calcule e mostre:

- A idade dessa pessoa em anos;
- A idade dessa pessoa em meses e;
- A idade dessa pessoa em dias.

10. Faça programa em C que receba 3 notas em avaliações de um determinado aluno e o peso de cada avaliação (3 pesos), calcule e mostre a média ponderada das notas do aluno.