

Aula 3: Construindo um programa em C

Professor(a): João Eduardo Montandon (103)

Virgínia Fernandes Mota (106)

[jemaf.github.io](https://github.com/jemaf)

<http://www.dcc.ufmg.br/~virginiaferm>

INTRODUÇÃO A PROGRAMAÇÃO - SETOR DE INFORMÁTICA



Agora que vocês tem uma noção de algoritmos básicos, vamos introduzir a linguagem de programação C:

- Estilo, leitura e usabilidade.
- Os elementos que contribuem para o estilo de um programa em C.
- Formas de se incluir código pré-escrito em um programa.
- Definição de constantes a serem usadas no programa.

Devemos adotar um estilo de escrita quando estamos programando?

Devemos adotar um estilo de escrita quando estamos programando?

Sim! O estilo vai depender de uma série de fatores:

- Se você for um estudante: o estilo vai facilitar que seu professor entenda seu código.
- Se você já trabalha: você terá que adotar o estilo imposto pela empresa.

A razão para adotarmos um estilo de programação está na facilidade de leitura

- Serão vários programas e vários programadores.
- Se todos usarem o mesmo estilo, todos irão ler e entender melhor os programas

O estilo universal

```
1 int main() {int a; a=20; printf("Imprimindo o valor de a: %d", a);  
    return 0;}
```

```
1 int main( ) {  
2     int a;  
3     a=20;  
4     printf("Imprimindo o valor de a: %d", a);  
5     return 0;  
6 }
```

Todo bloco de comandos deve ser indentado.

- Um programa bem feito não é apenas um conjunto de instruções.
- A documentação envolve todo texto que venha a descrever o que o programa faz.
- A documentação não deve ser feita após o programa estar pronto, mas sim à medida em que ele vai sendo desenvolvido.
- A documentação também deve conter o modo de uso do programa, bem como as simplificações assumidas pelo programador.

- Texto embutido no código-fonte, que ajuda no entendimento do programa.
- Regra geral: Sempre comentar trechos do programa que não são óbvias. Nunca comentar trechos do programa que sejam óbvios.
- Duas formas de inserir comentários:
 - 1 //linha comentada
 - 2 /*trecho comentado*/

- Geralmente uma operação bem mais complicada do que aquela que pode ser realizada por uma única instrução
- Uma função pode ser pré-estabelecida pelo próprio compilador ou criada pelo programador
- Após ser criada, uma função pode ser encarada como um comando criado pelo programador
- Uma função pode referenciar outras funções já criadas pelo programador

- O propósito de uma função deve ser bem definido.
- Toda função tem um nome. Esse nome é utilizado para acionar ou chamar a função
 - `printf("printf eh uma funcao");`
- O nome da função é sempre seguido de parênteses. Os parâmetros da função aparecem entre esses parênteses
 - `printf ("printf eh uma funcao");`
- Uma função pode conter nenhum, um, ou vários parâmetros. Parâmetros são sempre separados por uma vírgula.

Todo programa em C/C++ começa pela execução da função **main**.

- O programa começa pela chamada à função **main**.
- A partir daí, os comandos e funções inseridos em **main** são executados em sequência

Veremos como criar nossas próprias funções mais adiante no curso!

- Quando compilamos um programa, o compilador invoca um pré-processador, que associa um rótulo a um trecho de código ou a um valor.
- A diretiva **#include** <arquivo> instrue o compilador a inserir o trecho de código armazenado em *arquivo*
 - O arquivo onde está o trecho de código a ser incluído é chamado de arquivo cabeçalho.
 - Mais usados: **stdio.h**, **stdlib.h**, **math.h**
 - Cada arquivo contém trechos de código que implementam funções bem específicas
 - **math.h**: funções matemáticas
 - **stdio.h**: funções que imprimem e que lêem dados
 - **stdlib.h**: funções que implementam tarefas do sistema

- Valores que não mudam no decorrer da execução do programa podem ser representados por constantes.
- A declaração é dada pela diretiva **#define** *rótulo valor*
 - Exemplo: `#define PI 3.14`
 - O pré-processador substitui todas as referências a PI pelo valor 3.14
 - Geralmente o rótulo é definido em letras maiúsculas

é claro que podemos utilizar variáveis ao invés de constantes, porém a variável ocupa mais memória

Não é necessário fazer isto para qualquer constante!!!!!!

Apenas se for uma constante que será muito utilizada no código!

A saída de dados na tela pode ser realizada pela função **printf**. Existem códigos de conversão que nos permitem imprimir dados de diferentes tipos:

- `%c` → para imprimir dados do tipo caractere (*char*)
- `%d` ou `%i` → para imprimir dados do tipo inteiro (*int*)
- `%f` ou `%g` → para imprimir dados do tipo real (*float*)
- `%e` → para imprimir em notação científica
- `%s` → para imprimir dados do tipo cadeia de caracteres (*string*)

Exemplos

```
1 #include <stdio.h>
2 int main( ) {
3     printf("Estou aprendendo a programar em C");
4     return 0;
5 }
```

```
1 #include <stdio.h>
2 int main( ) {
3     printf("Valor recebido foi %d" ,10);
4     return 0;
5 }
```

Exemplos

```
1 #include <stdio.h>
2 int main(){
3     printf("Caracter A: %c" , 'A');
4     return 0;
5 }
```

```
1 #include <stdio.h>
2 int main() {
3     int x=10;
4     printf("Valor inteiro %d e um float %f" ,x ,1.10);
5     printf("Outro valor float %f", 1.10+2.35);
6     return 0;
7 }
```

Impressão de caracteres especiais

Código	Ação
<code>\n</code>	leva o cursor para a próxima linha
<code>\t</code>	executa uma tabulação
<code>\b</code>	executa um retrocesso
<code>\f</code>	leva o cursor para a próxima página
<code>\a</code>	emite um sinal sonoro (beep)
<code>\"</code>	exibe o caractere <code>"</code>
<code>\\</code>	exibe o caractere <code>\</code>
<code>%%</code>	exibe o caractere <code>%</code>

Fixando casas decimais

- Por **default**, a maioria dos compiladores C exibem os números de ponto flutuante (reais - float) com seis casas decimais.
- Para alterar este número podemos acrescentar .n ao código de formatação da saída, sendo n o número de casas decimais pretendido.

```
1 #include <stdio.h>
2 int main(){
3     printf("Default: %f \n", 3.1415169265);
4     printf("Uma casa: %.1f \n", 3.1415169265);
5     printf("Duas casas: %.2f \n", 3.1415169265);
6     printf("Tres casas: %.3f \n", 3.1415169265);
7     printf("Notacao Cientifica: %e \n", 3.1415169265);
8     return 0;
9 }
```

- O programa pode fixar a coluna da tela a partir da qual o conteúdo de uma variável, ou o valor de uma constante será exibido. Isto é obtido acrescentando-se um inteiro *m* ao código de formatação. Neste caso, *m* indicará o número de colunas que serão utilizadas para exibição do conteúdo.

```
1 #include <stdio.h>
2 int main() {
3     printf("Valor: %d \n",25);
4     printf("Valor: %10d \n",25);
5     return 0;
6 }
```

- 1. Fazer um programa que imprima o seu nome.
- 2. Modificar o programa anterior para imprimir na primeira linha o seu nome, na segunda linha a sua idade e na terceira sua altura.
- 3. Imprimir o valor 2.346728 com 1, 2, 3 e 5 casas decimais.

Exercícios

```
1 #include <stdio.h>
2 int main() {
3     printf("Meu nome");
4     return 0;
5 }
```

```
1 #include <stdio.h>
2 int main() {
3     printf("Meu nome \n Minha idade \n Minha altura");
4     return 0;
5 }
```

Exercícios

```
1 #include <stdio.h>
2 int main() {
3     printf(" %.1f", 2.346728);
4     printf(" %.2f", 2.346728);
5     printf(" %.3f", 2.346728);
6     printf(" %.5f", 2.346728);
7     return 0;
8 }
```

Operador de atribuição

- O operador de atribuição em C é o sinal de igual =
- Sintaxe: <variável> = <expressão>;

```
1 int a,b,c = 0,d;  
2 a = 5;  
3 b = a;  
4 d = a + b - c;  
5 a = (a/2)*3;  
6 a++; //similar a a = a + 1  
7 a--; //similar a a = a - 1  
8 a += 3; //similar a a = a + 3  
9 a -= 3; //similar a a = a - 3  
10 a /= 2; //similar a a = a/2  
11 a *= 2; //similar a a = a * 2
```

- Conversões automáticas de valores na avaliação de uma expressão.

```
1 int a, c;  
2 float b, d;  
3 a = 4.5; // conversão implícita  
4 b = a / 2.0; // conversão implícita  
5 c = 1/2 + b; // conversão implícita  
6 d = 1.0/2.0 + b;  
7 // Saída: a=4, b=2.0, c=2, d=2.5
```

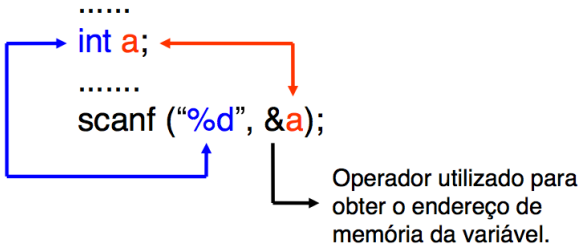
Conversão de Tipos - Cast

- é possível explicitamente fazer a conversão de tipos usando o operador **cast**.
- Sintaxe: **(tipo)** variável

```
1 int num1, num2, a;  
2 float resdiv;  
3 a = (int) 2.5;  
4 num1 = num2 = 5;  
5 resdiv =(float) num1 / num2;
```


Entrada - Leitura de Dados

- **scanf()**
- Ela é o complemento de **printf()** e nos permite ler dados formatados da entrada padrão (teclado).
- Sintaxe: `scanf("expressão de controle", argumentos);`
- Entendendo o **scanf()**:



Exemplos

```
1 #include <stdio.h>
2 int main() {
3     int num;
4     printf("Digite um valor: ");
5     scanf("%d",&num);
6     printf("\n O valor digitado foi %d.",num);
7     return 0;
8 }
```

Exemplos

```
1 #include <stdio.h>
2 int main() {
3     int n1, n2, soma;
4     printf("Digite dois valores: ");
5     scanf("%d", &n1);
6     scanf("%d", &n2);
7     soma = n1 + n2;
8     printf("\n %d + %d = %d.", n1, n2, soma);
9     return 0;
10 }
```

Pseudolinguagem para C

Exemplo 1: Ler uma temperatura em graus Celsius e apresentá-la convertida em graus Fahrenheit. A fórmula de conversão é $F = (9 \times C + 160) / 5$.

```
1 real C, F;
2 imprima("Digite uma temperatura em celsius");
3 leia(C);
4 F = (9*C + 160)/5;
5 imprima(C, "em celsius equivale a ", F, "em fahrenheit.");
```

```
1 #include <stdio.h>
2 int main() {
3     float C, F;
4     printf("Digite uma temperatura em celsius");
5     scanf("%f", &C);
6     F = (9*C + 160)/5;
7     printf(" %f em celsius equivale a %f em fahrenheit.", C, F);
8     return 0;
9 }
```

Pseudolinguagem para C

Exemplo 2: Calcular e apresentar o volume de uma lata de óleo cilíndrica, a partir da leitura do raio da base e da altura.

```
1 real R, H, V;  
2 imprima("Digite o raio da base e a altura");  
3 leia(R, H);  
4 V = 3.14*R*R*H;  
5 imprima("O volume é ", V);
```

```
1 #include <stdio.h>  
2 int main() {  
3     float R, H, V;  
4     printf("Digite o raio da base e a altura");  
5     scanf("%f %f", &R, &H);  
6     V = 3.14*R*R*H;  
7     printf("O volume é %f", V);  
8     return 0;  
9 }
```

Pseudolinguagem para C

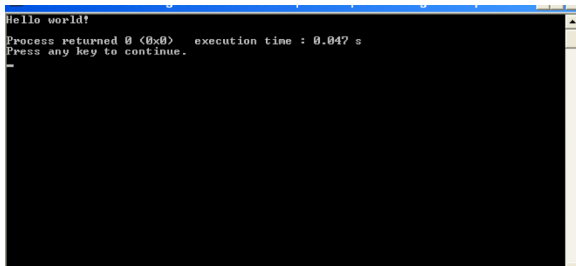
Exemplo 3: Ler os valores do comprimento, da largura e da altura de uma caixa, calcular e imprimir o seu volume.

```
1 real C, L, H, V;  
2 imprima("Digite o comprimento, a largura e a altura de sua caixa");  
3 leia(C, L, H);  
4 V = C*L*H;  
5 imprima("O volume é ", V);
```

```
1 #include <stdio.h>  
2 int main() {  
3     float C, L, H, V;  
4     printf("Digite o comprimento, a largura e a altura de sua caixa");  
5     scanf("%f %f %f", &C, &L, &H);  
6     V = C*L*H;  
7     printf("O volume é %f", V);  
8     return 0;  
9 }
```

Meu primeiro programa em C

```
1 #include <stdio.h>
2 int main() {
3     printf("Hello world!");
4     return 0;
5 }
```



A screenshot of a terminal window with a black background and white text. The text shows the output of a C program: "Hello world!" on the first line, "Process returned 0 (0x0) execution time : 0.047 s" on the second line, and "Press any key to continue." on the third line. The terminal window has a standard Windows-style title bar at the top.

```
Hello world!
Process returned 0 (0x0) execution time : 0.047 s
Press any key to continue.
```

1. Construir um algoritmo para ler 5 valores inteiros, calcular e imprimir a soma desses valores.
2. Construir um algoritmo para ler 6 valores reais, calcular e imprimir a média aritmética desses valores.
3. Fazer um algoritmo para gerar e imprimir o resultado do número H , sendo $H = 1 + 1/2 + 1/3 + 1/4 + 1/5$.
4. Calcular o aumento que será dado a um funcionário, obtendo do usuário as seguintes informações : salário atual e a porcentagem de aumento. Apresentar o novo valor do salário e o valor do aumento.
5. A nota final de um aluno é dada pela média ponderada das notas das provas. Sabendo que o professor deu 3 provas, com pesos 4, 3 e 3, respectivamente, calcule a nota final do aluno.

Na próxima aula...

Estruturas de Seleção