

Linguagem C Variáveis, Tipos de Dados, Comandos e Estrutura Linear

Prof. Edwar Saliba Júnior
Fevereiro de 2011

Curiosidade

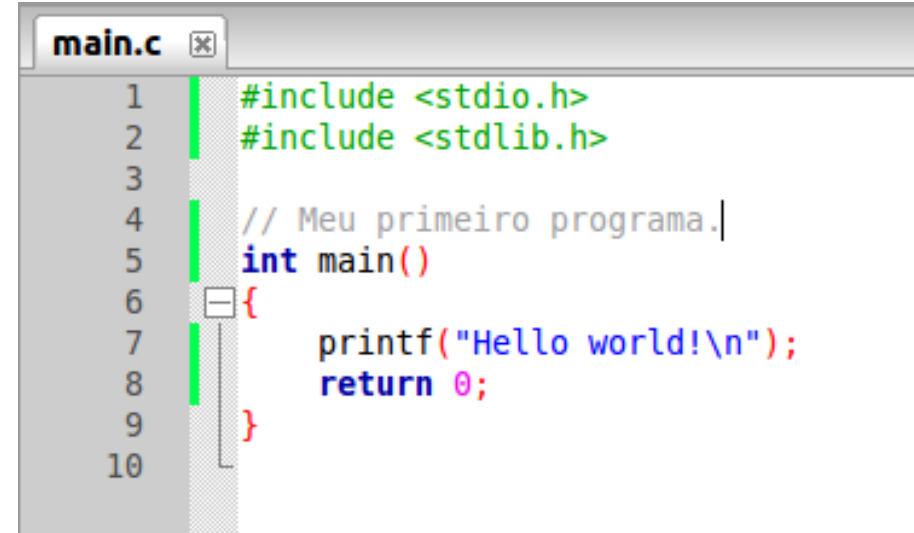
- A Pesquisa de Bohm e Jacopini (Deitel e Deitel, 2005, p. 88):
 - Demonstrou que todos os programas poderiam ser escritos com três estruturas de controle:
 - a estrutura de sequência;
 - a estrutura de seleção e
 - a estrutura de repetição.

Etapas da Programação

- Edição do programa:
 - Editor de texto (notepad, gedit e etc.) ou
 - IDE (*Integrated Development Environment*);
- Compilação:
 - Transformação do programa fonte em programa executável;
 - Feito pelo compilador: programa;
- Execução:
 - Feita pelo usuário usando o programa executável.

Programa em C

- Tudo em C é função;
- A função “*main*” é obrigatória. É por ela que um programa começa a ser executado;
- Todas as funções em C têm parêntesis;
- C é uma linguagem fracamente tipada;
- A linguagem C é *case sensitive*, ou seja, ela é sensível a caixa.
Ex.: *main* != *Main* != *mAin* !=
mAIIn != *MAIN*.



```
main.c [x]
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  // Meu primeiro programa.
5  int main()
6  {
7      printf("Hello world!\n");
8      return 0;
9  }
10
```

Variável

- O conceito de variável pode ser expresso de diversas maneiras:
 - Variável: armazena valores que podem variar;
 - Os dados (valores) que o programa utiliza, precisam ser armazenados na memória do computador antes de serem processados pela CPU: o lugar para isso são as variáveis;
 - Pedacos de memória utilizada pelos programas para armazenar e manipular valores.

Identificadores

- São as variáveis, as constantes, as funções e etc.;
- A cada qual será atribuído um nome, pelo programador;
- Regras para se definir os nomes dos identificadores:
 - Pode ser constituído por letras do alfabeto e/ou números de 0 a 9 e/ou o carácter “_” (*underscore*);
 - O primeiro carácter não pode ser um número;
 - Não se pode usar acentos;
 - Devem possuir nomes significativos.

Identificadores – Cuidados!

- O nome deve ser descritivo do conteúdo que o identificador armazena:
 - Correto: `double fatura; char resposta; int fatorial; float salario;`
 - Não: `double x; char a; int sdfg; float wxyz;`
- O nome não deve ser escrito todo em letras maiúsculas. Por convenção, as CONSTANTES em C, são escritas em maiúsculo;
- Escrever de forma a facilitar a leitura:
 - `nomeclienteprincipal;`
 - `NomeClientePrincipal;`
 - `nome_cliente_principal;`
 - `Nome_Cliente_Principal.`

Exemplos de Identificadores

- **int idade;** // ok!
- **int Num_Cliente;** // ok!
- **float x;** // ok!
- **float 5x;** // Erro: inicia com número.
- **double porcento%;** // Erro:carácter inválido %
- **char sim?nao;** // Erro: carácter inválido ?
- **int _alfa;** // ok!
- **char letra, Letra;** //duas variáveis diferentes.
Desaconselhável.

Tipos Básicos de Dados

- As variáveis e expressões podem assumir um destes tipos:

Tipo	Tamanho (bytes)	Valor
char	1	um caractere (ou um inteiro pequeno)
int	4	um número inteiro
float	4	número pequeno em ponto flutuante
double	8	número grande em ponto flutuante

Modificadores de Tipos

- São palavras que alteram o significado de um tipo base, definindo um novo tipo derivado:
 - As palavras são: *short*, *long*, *signed*, *unsigned*.
- Exemplos:
 - *short int*
 - *long int*
 - *signed char*
 - *signed int*
 - *signed short int*
 - *signed long int*
 - *unsigned char*
 - *unsigned int*
 - *unsigned short int*
 - *unsigned long int*

Tipos de Dados

Tipo	Número de Bits	Formato Leitura	Menor Valor	Maior Valor
char	8	%c	-128	127
unsigned char	8	%c	0	255
signed char	8	%c	-128	127
int	32	%i	-2147483648	2147483647
unsigned int	32	%u	0	4294967295
signed int	32	%i	-2147483648	2147483647
short int	16	%hi	-32768	32767
unsigned short int	16	%hu	0	65535
signed short int	16	%hi	-32768	32767
long int	32	%li	-2147483648	2147483647
unsigned long int	32	%lu	0	4294967295
signed long int	32	%li	-2147483648	2147483647
float	32	%f	3,40E-038	3,40E+038
double	64	%lf	1,7E-308	1,70E+308
long double	80	%Lf	3,4E-4932	3,4E+4932

- Essas informações podem ser encontradas no arquivo: `limits.h`

Declaração de Variáveis

- Toda variável precisa ser declarada antes de ser usada;
- Formato da declaração:
 - “tipo de dados” + “nome da variável”;
- Exemplos:

```
int contador;
```

```
char c;
```

```
float pi;
```

Atribuição

- Para atribuir um valor a uma variável, utilizamos o operador de atribuição: “=”

sintaxe: variável = expressão;

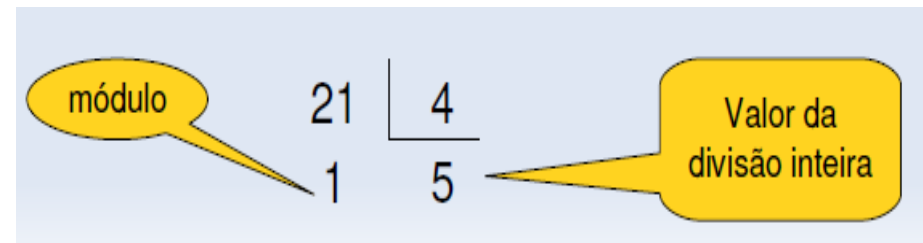
- Exemplos:

```
int num;  
num = 17;  
int numero = 17;  
int n1 = 3, n2 = 5;  
int a = 10, b, c, d = 123;  
int valor = num;  
a = b = c = d = 5;        // Atribuições múltiplas.
```

Operações com Números Inteiros e em Ponto Flutuante

- Operadores binários: + * / -
- Operadores unários: ++ --
- Resto da divisão inteira (módulo) %
- **Qualquer operação entre inteiros resulta em um inteiro;**
- Exemplo:

```
int a = 21 / 4; Retorna 5;  
int c = 21 % 4; Retorna 1;
```



- Outro Exemplo:

```
float b = 21 / 4; Retorna 5;  
float c = 21 / 4.0; Retorna 5.250000.
```

Operador Unário

- Operador unário ++ soma uma unidade à variável, o operador unário – subtrai uma unidade da variável;

- Exemplos:

`x++;` // equivalente a: `x = x + 1;`

`x--;` // equivalente a `x = x - 1;`

- Estes operadores podem ser pré-fixados ou pós-fixados:

`x = 23;` `y = x++;` // resulta em `y = 23` e `x = 24`

`x = 23;` `y = ++x;` //resulta em `y = x = 24`

- Pós-fixado: primeiro usa o valor na expressão, depois incrementa;
- Pré-fixado: primeiro incrementa, depois usa o valor na expressão.

Operadores Combinados

- As operações de incremento (++) e decremento (--) são exemplos de operações combinadas com a atribuição;
- Na linguagem C, sempre que for necessário escrever uma operação de atribuição da forma:

```
variável = variável operador expressão;
```

- poderemos combinar as operações.
- Exemplos:

```
x = x + 5;           x += 5;  
x = x - (a + b);    x -= (a + b);  
x = x * (a - b);    x *= (a - b);  
x = x / (x + 1);    x /= (x + 1);
```


Exercício

- Calcule o resultado de cada variável, após a execução das seguintes operações:

```
int x, y, z;
```

```
x = y = 10;
```

```
z = ++x;
```

```
x = -x;
```

```
y++;
```

```
z = z + x + y;
```

Tipo Char

- Variáveis deste tipo, são representadas pela linguagem como inteiros de 8 bits (um *byte*);
- Desta forma, também podem ser vistas também como inteiros;
- Cada carácter tem o valor inteiro de sua posição na tabela de ASCII (*American Standard Code for Information Interchange*);

```
main.c [x]
1  #include <stdio.h>
2
3  int main()
4  {
5      char Ch;
6      Ch='D';
7      printf ("\n %c",Ch); // Mostrará: D
8      printf ("\n %i",Ch); // Mostrará: 68
9  }
10
```

Formatação de Dados

- Utilizado principalmente pelas funções de entrada e saída de dados:

Código e Significado

<code>%i</code>	Inteiro (<code>int</code>)
<code>%d</code>	Decimal (<code>int</code>)
<code>%f</code>	Real (<code>float</code>)
<code>%c</code>	Carácter (<code>char</code>)
<code>%s</code>	<i>String</i> (<code>char[CONSTANTE]</code>)
<code>%e</code>	Número em notação científica
<code>%o</code>	Número no sistema octal
<code>%x</code>	Número no sistema hexadecimal
<code>%lf</code>	Real gigante (<code>Double</code>)

- Dica:

`%%` Escreve o carácter “%” na tela.

Comando `printf` (Saída de Dados)

- Sintaxe:

```
printf(string de controle, argumentos);
```

- **String de controle:** descreve tudo o que vai ser impresso na tela:
 - O texto a ser impresso;
 - Os valores das variáveis e em que posição;
- **Argumentos:** variáveis que terão seus valores impressos junto à *string* de controle;

- Exemplo:

```
int idadePedro = 15, idadeJoao = 14;
```

```
printf("Pedro tem %i anos e João tem %i.", idadePedro,  
idadeJoao);
```

- O local onde o valor de uma variável será impresso é marcado com o carácter especial "%".

Comando `scanf` (Entrada de Dados)

- A leitura de dados digitados pelo usuário no teclado, é feita através da função `scanf` com a seguinte sintaxe:

```
scanf (formatador, &argumento) ;
```

- **Formatador**: descreve o tipo de dados de entrada, que a função `scanf` espera receber;
- **&**: Retorna o endereço da variável;
- **Argumento**: variável que receberá os dados de entrada;
- Exemplo:

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

- Observação: sendo “idade” uma variável do tipo “int”.

Comando *gets* (Entrada de Dados)

- A leitura de dados do tipo *string* digitados pelo usuário no teclado, pode ser feita através da função `scanf`, contudo, esta função tem algumas limitações em se tratando de dados do tipo *string*. Para tal tarefa, temos a função `gets` com a seguinte sintaxe:

```
gets ( argumento ) ;
```

- **Argumento:** variável que receberá os dados de entrada;
- Exemplo:

```
gets (nome) ;
```

- Observação: sendo “nome” uma variável do tipo “char[xyz]”.

Exemplos de printf

- **Comando**

- `float valor = 1245.17;`
- `printf("O valor é %f", valor);`
- `int idade = 16;`
- `printf("Sua idade é %d anos.", idade);`
- `printf("Juros de 15%% ao ano.");`
- `char letra = 'N';`
- `printf("O carácter é o %c.", letra);`
- `printf("O carácter é o %d.", letra);`
- `printf("O carácter é o %o.", letra);`
- `printf("O carácter é o %x.", letra);`

Saída na Tela

O valor é 1245.17

Sua idade é 16 anos.

Juros de 15% ao ano.

O carácter é o N.

O carácter é o 78.

O carácter é o 116.

O carácter é o 4E.

Exemplos de *scanf* e *gets*

• Digitado	Comando	Armazenado
•	<code>float valor;</code>	
• 1245.17	<code>scanf("%f", &valor);</code>	1245.17
•	<code>int idade;</code>	
• 16	<code>scanf("%d", &idade);</code>	16
• 112.34	<code>scanf("%d", &idade);</code>	112
•	<code>char letra;</code>	
• T	<code>scanf("%c", &letra);</code>	T
•	<code>char nome[30];</code>	
• Paulo César	<code>scanf("%s", &nome);</code>	Paulo
• Paulo César	<code>gets(nome);</code>	Paulo César

Caracteres Especiais: Constantes Iniciadas por \

- **Código** **Significado**
- \ b Retrocesso ("*back*")
- \ f Alimentação de formulário ("*form feed*")
- \ n Nova linha ("*new line*")
- \ t Tabulação horizontal ("*tab*")
- \ " Aspas
- \ ' Apóstrofo
- \ 0 Nulo (0 em decimal)
- \ \ Barra invertida
- \ v Tabulação vertical
- \ a Sinal sonoro ("*beep*")
- \ N Constante octal (N é o valor da constante)
- \ xN Constante hexadecimal (N é o valor da constante)

Programa em C

- Exemplo de uso das estruturas de entrada e saída de dados:

```
main.c [x]
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main()
5  {
6      int idade;
7      float altura;
8      char nome[15];
9
10     printf("Digite seu nome: ");
11     gets(nome);
12     printf("\nDigite sua idade: ");
13     scanf("%i",&idade);
14     printf("\nDigite sua altura: ");
15     scanf("%f",&altura);
16
17     printf("\n\nSeu nome é %s, você mede %f mts e tem %i anos.", nome, altura, idade);
18     return 0;
19 }
20
```

Formatando a Impressão

- É possível estabelecer o tamanho mínimo para impressão de um valor:

```
main.c [x]
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main()
5  {
6      printf("São %2d alunos.\n", 350);
7      printf("São %4d alunos.\n", 350);
8      printf("São %5d alunos.\n", 350);
9
10     return 0;
11 }
12
```

```
FormatacaoPrintf
São 350 alunos.
São  350 alunos.
São   350 alunos.

Process returned 0 (0x0)   execution time : 0.010 s
Press ENTER to continue.
```

`%2d` significa que será impresso, um número decimal com no mínimo duas posições.

Formatação

- Inteiros

Valor	Formatação	Valor exibido
3	%d	3
	%5d	3
	%01d	3
	%05d	00003

- Reais

Valor	Tag	Valor exibido
pi = 3.14159	%5.3f	3.142
	%8.3f	3.142
raio = 2.0031	%5.3f	2.003
	%.6f	2.003100
area = 2*pi*raio	%5.3f	12.586
	%6.3f	12.586
	%7.3f	12.586
	%e	1.258584e+001
	%E	1.258584E+001
	%12.3e	1.259e+001

Exercícios

- Faça um programa para determinar a quantidade de litros de combustível gastos em uma viagem, por um automóvel que faz 12 km/litro. Para isto, sabe-se que o tempo gasto na viagem é $T = 35$ min e a velocidade média do automóvel é $V = 80$ km/h.
- Faça um programa para ler do teclado, dois valores inteiros, A e B. Este programa deverá efetuar a troca dos valores, de forma que a variável A passe a possuir o valor da variável B, e vice-versa. Apresente os valores das duas variáveis.

Bibliografia

- DEITEL, H. M.; DEITEL, P. J. **Java Como Programar**; tradução Edson Furmankiewicz; revisão técnica Fábio Lucchini. 6a. ed., São Paulo: Pearson, 2005.
- MURTA, Cristina Duarte. *Slides* da disciplina de Programação de Computadores I. CEFET-MG, 2010.
- SENNE, Edson Luiz França. **Primeiro Curso de Programação em C**. 2. ed. Florianópolis: Visual Books, 2006.