COMANDO DE ATRIBUIÇÃO

Ao declarar uma variável, o compilador reserva espaço na memória para o **armazenamento de valor**.

Como a memória do computador é composta por bytes, formados a partir de bits, a variável pode assumir um valor aleatório, uma vez que não temos controle sobre eles.

A inicialização só ocorre quando se atribui valor por meio de um **comando de atribuição**. A seguir, vamos ver um pouco mais sobre o uso dessa ferramenta na programação.

Todos os comandos apresentados aqui obedecem à sintaxe (conjunto de regras) da **linguagem C** e do **Portugol**. Antes de utilizar o comando de atribuição, você deve **inicializar** a variável.

Tomemos como exemplo a declaração da variável inteira chamada **a**. Vejamos como realizar esse procedimento:

***int* a;**

Na linguagem C e no Portugol, esse comando é representado pelo sinal de igual **=**, conforme se observa no formato geral da estrutura:

nome_da_variável = valor_atribuído;

Após a declaração de **a**, existem duas maneiras de atribuir o valor 10 a essa variável:

***int* a;**

a = 10;

ou

***int* a = 10;**

O nome da variável deve ajudar a entender seu significado.

O uso de iniciais maiúsculas, a partir da segunda palavra, ou do símbolo underscore _ permite a criação de nomes mais complexos, como: idCliente, id_cliente, cpf_usuario, cpfUsuario, entre outros.

Na linguagem C, ainda é possível atribuir o mesmo valor a mais de uma variável. Com a seguinte instrução, é dado o valor 2 às variáveis **a** e **b**. Vejamos um exemplo:

a = b = 2;

Observe que não há como guardar o histórico de valores de uma variável. A atribuição de outro valor faz com que o anterior seja perdido. Para evitar que isso aconteça, deve-se usar outra variável. Na sequência de instruções a seguir, a variável **a** vale 3, sem que 1 e 2 sejam guardados. Observe:

***int* a;**

a = 1;

a = 2;

a = 3;

Em **pseudocódigo**, o comando de atribuição é representado pela seta (←), mas não simboliza a igualdade; ele atribui à variável do lado esquerdo o valor que está à direita. Vejamos alguns exemplos:

Exemplo

a ← 10 (pseudocódigo) ou a = 10 (Portugol e C) atribui o valor 10 à variável a.
a ← a + 1 (pseudocódigo) ou a = a +1 (Portugol e C) acresce uma unidade à variável a, resultando no valor 11.

O mesmo ocorre na próxima sequência, em que **a** teria o valor 6 ao final da execução das instruções:

a ← 5 (pseudocódigo) ou a = 5 (Portugol e C).
a ← a + 1 (pseudocódigo) ou a = a +1 (Portugol e C).

O comando de atribuição pode ser usado para variáveis dos tipos **int**, *double* e *float* da mesma forma que vimos anteriormente. Por outro lado, o tipo **char** deve ser usado com cautela para que não haja confusão entre o uso de caractere e variável, conforme é mostrado a seguir.

Para declarar uma variável do tipo **char** chamada **escolha**, usamos:

char **escolha;**

Como é do tipo **char**, espera-se receber caracteres. Para atribuir **b** à escolha, utilizaremos as aspas simples a fim de indicar que se trata do caractere **b**, e não da variável **b**, sendo o comando correto:

escolha = 'b';

Caso seja feito sem as aspas simples, o programa apontará erro, já que o compilador irá procurar a variável **b**, não declarada, para atribuir o seu valor à **escolha**.

A linguagem C também permite operações aritméticas com variáveis do tipo char, relacionando o valor dos caracteres armazenados nelas aos inteiros correspondentes na tabela ASCII.

No ASCII, existem apenas 95 caracteres que podem ser impressos. Eles são numerados de 32 a 126, sendo os caracteres de 0 a 31 reservados para funções de controle. Veja alguns caracteres especiais:

\7	*Bell* (sinal sonoro do computador)	\	Caractere \ (forma de representar o próprio caractere especial \ )
\a	*Bell* (sinal sonoro do computador)	**\'**	Caractere ' (aspas simples)
\b	*BackSpace*	**\"**	Caractere " (aspas)

\n	*New Line* (mudança de linha)	**\?**	Caractere ? (ponto de interrogação)
\r	*Carriage Return*	**\000**	Caractere cujo código ASCII em Octal é **000**
\t	Tabulação Horizontal	**\xyy**	Caractere cujo código ASCII em Hexadecimal é **yy**
\v	Tabulação Vertical		

Tabela: Caracteres especiais.
Humberto Henriques de Arruda

O próximo exemplo ilustra essa relação. Veja:

char **escolha;**

escolha = ‘b’;

escolha = escolha + 1;

Ao final da execução dessas linhas, a variável escolha armazenará o caractere ‘c’.

Vamos praticar

Você receberá agora uma série de práticas para realizar em seu ambiente de programação. Tente executá-las. Vamos lá!

Prática 1

Vamos descobrir qual é o valor da variável cont após a execução das seguintes linhas:

#include <stdio.h>

int **main(){**

int **cont = 1;**

cont = cont + 1;

```
printf("A variável cont ao final possui o valor: %d ", cont);
return 0;
}
```

O valor da variável é 2. A vari**ável cont** é inicializada com 1, mas a segunda linha acresce uma unidade a esse valor.

Prática 2

Vamos descobrir qual é o valor da variável **escolha** após a execução das seguintes linhas:

```
#include <stdio.h>
int main(){
char escolha;
escolha = 'D';
escolha = escolha - 2;
printf("A variável escolha possui o valor: %c ", escolha);
return 0;
}
```

O valor da variável é '**B**'. Por se tratar de um caractere, ao realizar a operação aritmética para diminuir duas unidades da variável escolha, ficará aquele que estiver duas posições antes na tabela ASCII (nesse caso, no alfabeto). Vale lembrar que a linguagem C é *case sensitive*, ou seja, diferencia letras maiúsculas de minúsculas.

Prática 3

Vamos descobrir qual é o valor da variável c após a execução das seguintes linhas:

```
#include <stdio.h>
int main(){
int a, b, c, d;
a = 10;
```

```
b = a + 1;
c = b + 1;
d = c + 1;
a = b = c = d = 20;
printf("A variável c possui o valor: %d ", c);
return 0;
}
```

O valor da variável é 20. A última linha atribui valor 20 a todas as variáveis, não importando o valor que tinham previamente.

Prática 4

Vamos descobrir qual é o resultado da execução das seguintes linhas:

```
#include <stdio.h>
int main(){
char escolha;
escolha = a;
escolha = escolha + 1;
printf("A variável escolha possui o valor: %d ", escolha);
return 0;
}
```

Ocorrerá erro de compilação na segunda linha por não haver variável declarada com o nome **a**. Lembre-se sempre de não confundir caractere ‘**a**’ com variável **a**.