Professor
Edson Pedro FERLIN

Programação em C

Variáveis, Constantes, Operadores e Expressões

Prof. Edson Pedro Ferlin

Professor
Edson Pedro FERLIN

Programação em C

Tipos de Dados
Básicos

- Cinco (5) tipos básicos:
  - **char** (caracter);
  - **Int** (Inteiro);
  - **float** (ponto flutuante);
  - **double** (ponto flutuante de dupla precisão);
  - **void** (sem valor);

Tipo	Tamanho (bits)	Escala
char	8	-128 a 127
int	16	-32.768 a 32.767
float	32	3,4E-38 a 3,4E+38
double	64	1,7E-308 a 1,7E+308
void	0	Sem valor

Tipos de Dados
Modificadores de Tipo

- Modificadores de tipo:
  - **signed** (com sinal);
  - **unsigned** (sem sinal);
  - **long** (longo);
  - **short** (curto);
- São utilizados para alterar o significado do tipo base, p.ex:
  - **unsigned char** (0 – 255)
  - **long int** (-2.147.483.648 a 2.147.483.647)
  - **unsigned int (**0 a 65.535)

Obs: **unsigned** e **long** sozinhas representam tipo **int.**

Variáveis
Visão Geral

- Duas condições:
  - O nome deve começar com uma letra ou sublinhado (_);
  - Os caracteres subsequentes devem ser letras, números ou sublinhado (_);
- Duas restrições:
  - O nome de uma variável não pode ser igual a uma palavra reservada;
  - O nome não pode ser igual ao nome de uma função declarada pelo programador, ou pelas bibliotecas do C;
- Variáveis de até 32 caracteres são aceitas;
- Case sensitive (maiúsculas e minúsculas);

Dicas:

Usar letras minúsculas para nomes de Variáveis;
Usar letras maiúsculas para nomes de Constantes;

Variáveis
Declaração

Forma geral de um comando de declaração de variáveis:

[modificador] tipo_da_variável Lista_de_Variáveis;

Exemplo:

- int i,j,l;
- short int si;
- float pi;
- char ch;
- long count;

Variáveis
Escopo

- Existem três lugares em um programa em C onde podemos declarar variáveis:
  - ***Global***: Fora de todas as funções;
  - ***Local***: Dentro de uma função;
  - ***Parâmetro formal***: declaração dos parâmetros;

Variáveis
Exemplo de Programa

```
int sum;              /* Variavel Global */
main ()
{
    int count;        /* Variavel Local */
    sum = 0;
    for (count = 0; count < 10; count ++)
    {
        total (count);
        display ();
    }
}
```

```
total (x)             /* Parametro formal */
int x;
{
    sum = x + sum;
}

display ()
{
    int count;
    for (count = 0; count < 10; count++)
    {
            printf (".");
    }
    printf ("A soma corrente eh %d\n",sum);
}
```


Variáveis
Observações

- Observações a serem seguidas:
  - Duas variáveis globais não podem ter o mesmo nome;
  - Variáveis locais em uma função pode ter o mesmo nome de outras variáveis locais em outra função, sem conflitos;
  - Duas variáveis dentro de uma mesma função não podem ter o mesmo nome.

Constantes

- São valores que são mantidos fixos pelo compilador;
  - Tipos básicos: ‘b’, ‘\o’, 2, -467, 0.0
  - Hexadecimais e octais: 0xFF (255), 011(9)
  - Strings: “Teste”,”t”
  - Barra invertida: \n, \b

Inicialização de Variáveis

Forma Geral

- Podemos dar valores às variáveis no momento em que se declara, colocando um sinal de igual e uma constante depois do nome da variáveis;
- Forma geral:

  Tipo Nome_da_Variável = constante;

- Exemplos:
  - int hex=0xFF;
  - char CH=‘A’;
  - int ANO=2011;
  - float balanço = 123.4;

Inicialização de Variáveis

Globais e Locais

- Variáveis Globais → inicializadas apenas no começo do programa. São inicializadas em zero, se não for especificado;
- Variáveis Locais → inicializadas cada vez que a função na qual elas são declaradas é inserida. As que não forem inicializadas terão valores desconhecidos antes que seja feita a primeira atribuição.
- Vantagem: Reduz sensivelmente a quantidade de código do programa;

Exemplo

```
main ()
{
     int t;                /* Variavel Local */
     printf (“Digite um número: “);
     scanf (“%d”, &t);
     total (t);
}
total (x)                  /* Parametro formal */
int x;
{
   int soma = 0, i, cont;
   for (i = 0; i < x; i++)
   {
        soma = soma + 1;
        for (cont = 0; cont < 10; cont++) prinft (“-”);
   }
}
```

Operadores
Classes

- Um operador é um símbolo que manda o compilador executar determinadas manipulações matemáticas ou lógicas;
- Os operadores em C podem ser agrupados nas classes:
  - Operadores Aritméticos
  - Operadores Relacionais
  - Operadores Lógicos
  - Operadores *bit-a-bit*
  - Operadores de Atribuição
  - Operador de Tamanho
  - Operador Vírgula

Operadores
Aritméticos

- Os operadores aritméticos são:

-	Subtração, também menos unário;
+	Adição;
*	Multiplicação;
/	Divisão;
%	Resto da divisão;
--	Decremento;
++	Incremento.

Precedência
++, --
- (unário)
* ,/, %
+, -

- O compilador avalia os operadores no mesmo nível de precedência da esquerda para a direita.

Operadores
Relacionais e Lógicos

- Os operadores relacionais e lógicos são:
  - > Maior que;
  - >= Maior ou igual a;
  - < Menor que;
  - <= Menor ou igual a;
  - == Igual a;
  - != Não igual a;
  - && AND ;
  - || OR;
  - ! NOT.

Precedência
!
>, >=, <, <=
==, !=
&&
||

- As expressões que usam operadores relacionais e lógicos retornarão “0” para Falso e “1” para verdadeiro.

Operador
de Atribuição

- Os operadores de Atribuição são:
  - *a= b=c; significa a = b; a=c;*
  - *a += b; significa a = a + b;*
  - *a -= b; significa a = a - b;*
  - *a *= b; significa a = a * b;*
  - *a /= b; significa a = a / b;*
  - *a=b++ significa a = b; b=b+1;*
  - *a=++b significa a = b+1; b=b+1;*
  - *a=b-- significa a = b; b=b-1;*
  - *a=--b significa a = b-1; b=b-1;*

Precedência
++,--
=, +=,-=,*=,/=

Operadores
de Bit a Bit

- Os operadores de Bit a Bit são:
  & AND;
  || OR;
  ^ XOR (OR – Exclusivo);
  ~ NOT;
  >> Deslocamento de bits à direita;
  << Deslocamento de bits à esquerda.

  - Exemplo: i << 3; desloca 3 bits para a esquerda.

Operadores
de Tamanho e Vírgula

- O **operador de tamanho** é obtido por meio do operador sizeof(), que fornece a quantidade de bytes de seu operando:

int x;	x = sizeof (int);	fornece 2 bytes
float a;	x = sizeof a;	fornece 4 bytes
int b;	x = sizeof b;	fornece 2 bytes
char c;	x = sizeof c;	fornece 1 byte

- O **operador vírgula** é usado para unir expressões relacionadas. Uma lista de expressões separadas por vírgulas é tratada como uma expressão única e avaliada da esquerda para a direita.

  if (c = getchar() , c > 'a')

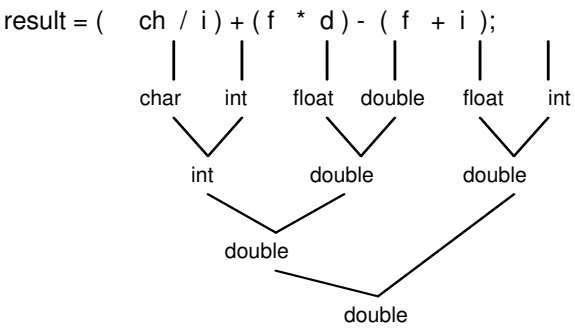
Expressões
Conversão de Tipos

- Quando constantes, variáveis e funções de tipos diferentes são misturados em uma expressão, elas são convertidas para o mesmo tipo. *O compilador C converterá todos para o tipo do operando maior*. Isto é feito operação a operação, seguindo as regras :

  Todos os chars e short ints são convertidos para int. Todos os floats são convertidos para doubles.

  Para todos os pares de operandos, se um deles e um *long double*, o outro operando é convertido para um *long double*. Se um é *double*, o outro é convertido para *double*. Se um é *long*, o outro é convertido para *long*. Se um é *unsigned*, o outro é convertido para *unsigned*.

Expressões
Conversão de Tipos - Exemplo



Expressões
Modelador (Cast)

- É possível *forçar uma expressão a ser de um tipo* específico usando-se uma construção chamada de modelador (cast).
- A forma geral de um modelador é:

(tipo) expressão

Se x é um inteiro:

(float) x / 2 resultará em x/2 float, pois o modelador força a operação no tipo float.

(float) (x / 2) resultará primeiramente x/2 é calculada como inteiro, depois o resultado da operação será devolvido como float.

Expressões
Espaçamento e Parênteses

- Pode-se colocar espaços numa expressão para torná-la mais legível.
- O uso de parênteses redundantes ou adicionais não causará erros ou diminuirá a velocidade de execução da expressão.

```
a=b/9.67-56.89*x-34.7;

a = ( b / 9.67 ) -  ( 56.89 * x ) - 34.7 ;
```

Contato

eferlin@live.com

(BLOG) professorferlin.blogspot.com

(SITE) professorferlin.webnode.com.br

(YOUTUBE) ProfEdsonPedroFerlin