MC102 - Algoritmos e Progração de Computador

Prof. Alexandre Xavier Falcão

2° Aula: Variáveis simples, atribuições e operações matemáticas

1 Variáveis simples

Variáveis simples armazenam números e caracteres na memória principal. A forma de armazenamento depende do conteúdo. Variáveis que armazenam números inteiros, por exemplo, consideram 1 bit para o sinal e os demais para o valor do número. Por exemplo, uma variável inteira com b bits pode armazenar valores de $-2^{(b-1)}$ a $2^{(b-1)}-1$ ou valores sem sinal de 0 a 2^b-1. Essas variáveis são declaradas da seguinte forma.

```
int main()
{
int a; /* pode armazenar apenas números inteiros com no máximo 4 bytes,
  incluindo o sinal. Isto é, valores de -2147483648 a 2147483647. */

unsigned int b; /* pode armazenar apenas números inteiros sem sinal com no
     máximo 4 bytes. Isto é, valores de
     0 a 4294967295. */

short c; /* pode armazenar apenas números inteiros com no máximo 2 bytes,
    incluindo o sinal. Isto é, valores de -32768 a 32767. */

unsigned short d; /* pode armazenar apenas números inteiros sem sinal
          com no máximo 2 bytes. Isto é, valores de 0 a
          65535. */

float e; /* pode armazenar números reais com no máximo 4 bytes,
            incluindo o sinal. A forma de armazenamento permite
            representar números nos intervalos de -2 x 10^(-38) a -2 x
            10^(38) e de 2 x 10^(-38) a 2 x 10^(38). */

double f; /* pode armazenar números reais com no máximo 8 bytes,
            incluindo o sinal. A forma de armazenamento permite
            representar números nos intervalos de -2 x 10^(-308) a -2 x
            10^(308) e de 2 x 10^(-308) a 2 x 10^(308). */

char   g; /* pode armazenar caracteres alfanuméricos com no máximo 1
```

```
  byte, incluindo o sinal no caso de número inteiro. Isto é,
  valores como 'a', 'X', '%' e números de -128 a 127.*/

unsigned char   h; /* pode armazenar caracteres alfanuméricos com no máximo 1
  byte, incluindo o sinal no caso de número inteiro. Isto é,
  valores como 'a', 'X', '%' e números de 0 a 255.*/

}
```

Alguns sistemas operacionais consideram apenas 2 bytes para o tipo int. Neste caso, o tipo **long** estende para 4 bytes. O comando **sizeof(tipo)** permite saber quantos bytes são ocupados um dado **tipo** de variável (e.g. $sizeof(int) = 4$ bytes).

Os nomes das variáveis podem ter tamanhos maiores, contendo caracteres alfanuméricos, mas não podem conter caracteres e nem constituírem palavras reservadas pela linguagem para outros fins.

2 Atribuições

O operador "=" é usado para indicar uma atribuição de valor à variável. Todo comando de declaração de variável ou de atribuição deve ser finalizado com ";". Exemplos:

```
int main
{
int a;
unsigned int b;
short c;
unsigned short d;
float e;
double f;
char   g;
unsigned char h;

a = 10; /* correto */
b = -6; /* errado */
c = 100000; /* errado */
d = 33000; /* certo */
e = -80000.657; /* certo */
f = 30 /* errado */
g = 'a'; /* certo */
g = a; /* errado, a menos que ``a'' fosse do tipo char */
h = 200; /* certo */
h = 'B'; /* certo */
}
```

Declarações e atribuições também podem aparecer da seguinte forma:

```
int main()
{
```

```
int a=10,b=-30; /* na mesma linha, separadas por ‘‘,’’. */
float c;
char d='4'; /* estou me referindo ao caracter 4 e não ao número. */

c = a; /* converte para float e copia valor 10.0 para ‘‘c’’. */
c = a + 1.8; /* atribui valor 11.8 para ‘‘c’’. */
b = c; /* converte para int truncando a parte decimal e copia 11
  para ‘‘c’’. */
b = a + b; /* soma 10 e 11, e copia 21 para. b */
a = 10 + b; c = b*40.5; /* devemos evitar de escrever vários comenados
   em uma mesma linha. */
}
```

Note que as atribuições só podem ser feitas após a declaração da variável. Podemos também usar um **tipo** de variável entre parênteses, para fazer com que o resultado da expressão seja convertido para o tipo especificado antes da atribuição. Exemplos:

```
int main()
{
int a;

a = (int)(2.5*7.2); /* calcula o produto e depois converte para inteiro. */
}
```

3 Operações aritméticas e expressões

Operações aritméticas podem ser combinadas formando expressões. Existe uma regra de prioridade da multiplicação e divisão (mesma prioridade) sobre soma e subtração (mesma prioridade), e a ordem de execução para operadores de mesma prioridade é da esquerda para direita. Exemplos:

```
int main()
{
  int a=20,b=10;
  float c=1.5,d;

  d = c*b/a; /* atribui 0.75 para ‘‘d’’ */
  d = c*(b/a); /* atribui 0.0 para ‘‘d’’, pois a divisão entre
    inteiros resulta em um inteiro. A divisão só resulta
    em número real se ao menos um dos operandos for
    real, porém isto não adianta nada se o resultado for
    atribuído a uma variável inteira. Os parênteses
    forçam a execução da operação de divisão antes da
    multiplicação. */
  d = b-a*c; /* atribui -20.0 para ‘‘d’’ */
  d = (b-c)*a; /* atribui 170.0 para ‘‘d’’ */
}
```

Note que a melhor forma de garantir o resultado esperado é usar parênteses. A prioridade de execução neste caso é da expressão mais interna para a mais externa.

```
int main()
{
  int a=20,b=10;
  float c=1.5,d;

  d = (((a+5)*10)/2)+b; /* atribui 135 para ‘‘d’’ */
}
```

4 Exercício

Sabendo que as fórmulas de conversão de temperatura de Celsius para Fahrenheit e vice-versa são

$$C = \frac{(F-32)5}{9}, \quad (1)$$

$$F = \frac{9C}{5} + 32, \quad (2)$$

escreva dois programas em C, um para converter de Celsius para Fahrenheit e outro para fazer o caminho inverso.