MC-102 Algoritmos e Programação

de Computadores
IC-UNICAMP

Aula 04 - Variáveis (Continuação)

Por: Luís Augusto Angelotti Meira
(Sala IC-71) 1S2005

1 Objetivos

Explorar a função `printf` e `scanf`, alinhamento, operador de endereço, operadores aritméticos `= + - * / %`, precedência e `()`, operadores de incremento e decremento `++ --`, operadores aritméticos de atribuição `+= -= *= /= %=`, operadores relacionais `> < >= <= == !=`, construção de programas simples.

2 Motivação

Utilização de `printf`, `scanf` e operadores permite a construção de programas simples.

3 Aula e Exemplos

A aula que se segue baseou-se em [1, 3, 2].

3.1 Definição Formal de Variáveis

Uma variável é composta por um nome, um endereço e um conteúdo.

3.2 Explorando a Função `printf`

Para inteiros temos que o seguinte programa

```
#include <stdio.h>
int main(void){
  printf("Os alunos são %d.\n",350);
  printf("Os alunos são %2d.\n",350);
  printf("Os alunos são %4d.\n",350);
  printf("Os alunos são %6d.\n",350);
  printf("Os alunos são %02d.\n",350);
  printf("Os alunos são %04d.\n",350);
  printf("Os alunos são %06d.\n",350);
```

```
    printf("Os alunos são %10.02d.\n",350);
    printf("Os alunos são %10.04d.\n",350);
    printf("Os alunos são %10.06d.\n",350);

  }
```

terá como saída

```
Os alunos são 350.
Os alunos são 350.
Os alunos são  350.
Os alunos são    350.
Os alunos são 350.
Os alunos são 0350.
Os alunos são 000350.
Os alunos são        350.
Os alunos são       0350.
Os alunos são     000350.
```

O seguinte programa :

```
#include <stdio.h>
int main(void){
printf("%f %f\n%f %f\n\n",3456.78 ,  56.78  , 6.78  , 0.78  );
printf("%10.1f %10.1f\n%10.1f %10.1f\n\n",3456.78 ,  56.78  , 6.78  , 0.78  );
printf("%-10.3f %-10.3f %-10.3f \n%-10.3f %-10.3f %-10.3f\n\n",
         3456.78 ,  56.78 , .1 , 6.78  , 0.78 ,100.23334 );
}
```

terá como saída

```
3456.780000 56.780000
6.780000 0.780000

    3456.8       56.8
       6.8        0.8

3456.780   56.780     0.100
6.780      0.780      100.233
```

3.3 A função `scanf`

A função `scanf` é uma função de leitura. Ela recebe como parâmetros um ou mais tipos e uma ou mais variáveis. Ela, durante a execução do programa, aguarda que o usuário digite um valor e atribui o valor digitado à variável correspondente. Ex:

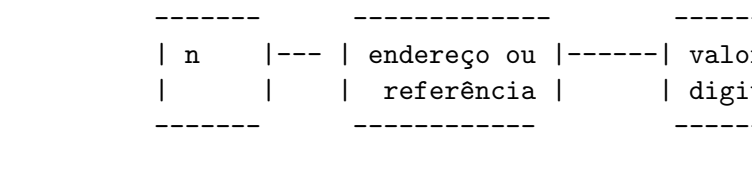
```
#include <stdio.h>
int main(void){
  int n;
  printf("Digite um número e pressione Enter: ");
  scanf("%d",&n);
  printf("O valor digitado foi %d\n",n);
}
```

O programa acima é composto de quatro passos:

1. Cria uma variável `n`;
2. Escreve na tela `Digite um número:`
3. Lê o valor do número digitado
4. Imprime o valor do número digitado

3.4 O operador "address-of" & de C

A variável `n` do exemplo acima, tem associada a ela um endereço de memória.




```
Obs. O valor da variável n  ter que ser do tipo inteiro, lembrando que
a variável também tem um tipo de dados associado a ela.
```

Normalmente, o endereço de variáveis não são conhecidos quando o programa é escrito. O endereço de uma variável é dependente do sistema computacional e também da implementação do compilador C que está sendo usado, sendo que o endereço de uma mesma variável pode mudar entre diferentes execuções de um mesmo programa C usando uma mesma máquina.

Portanto, é útil para um programador a disponibilidade de operadores e expressões que o auxiliem na obtenção do endereço corrente de uma variável. O operador "address-of" & é um exemplo.

Quando o operador & é aplicado no nome de uma variável (por exemplo, &n), ele retorna o endereço corrente da variável `n`. Portanto, se `n` é uma variável inteira, &n é um endereço que referencia `n`.

É necessário usar o operador `&` no comando `scanf`, pois esse comando coloca o valor digitado na tela, no endereço referente à variável `n`, e não na variável `n`.

Esquecer de colocar o `&` comercial é um erro muito comum e somente vai ser percebido na execução.

O programa abaixo imprime o valor e o endereço da variável:

```
#include <stdio.h>
int main(void){
  int n = 8;
  printf("valor %d, endereço %u\n",n,&n);
}
```

Um exemplo de saída é

```
valor 8, endereço 3221217828
```

A tabela seguinte têm os tipos básicos aceitos pela função `scanf`:

Código	Função
%c	Lê um único caracter
%d	Lê um número decimal
%e	Lê um número em notação científica
%f	Lê um número em ponto flutuante
%o	Lê um inteiro octal
%s	Lê uma série de caracteres
%x	Lê um número hexadecimal
%u	Lê um decimal sem sinal
%l	Lê um inteiro longo
%lf	Lê um double

Mais um exemplo do uso do `scanf`

```
#include <stdio.h>
int main(void){
  float anos, dias;
  printf("digite sua idade em anos: ");
  scanf("%f",&anos);
  dias = anos * 365 ;
  printf("Sua idade em dias é %.0f.\n",dias);
}
```

Eis uma execução do programa

```
digite sua idade em anos: 22.13
Sua idade em dias é 8077.
```

3.5 Operadores Aritméticos

Binários	
=	Atribuição
+	Soma
-	Subtração
*	Multiplicação
/	Divisão
%	Módulo (resto da divisão inteira)

Unário	
-	Menos unário

Dar exemplos de cada um dos operadores.

3.6 Operador incremento e decremento ++ --

Explicar cada uma das atribuições:

- `n++;`
- `++n;`
- `n--;`
- `--n;`
- `a = b++;`
- `a = ++b ;`
- `a = b--;`
- `a = --b;`
-

Comando	Exemplo	Corresponde a:
`+=`	`a += b`	`a = a + b;`
`-=`	`a -= b`	`a = a - b;`
`=`	`a *= b;`	`a = a * b;`
`/=`	`a /= b;`	`a = a / b;`
`%=`	`a %= b;`	`a = a % b;`

3.7 Precedência

Uma expressão como a + b * c será calculada como a + (b * c) porque o cálculo da mesma leva em conta a precedência entre os operadores. Em C essa precedência é definida da seguinte forma (para os operadores aritméticos):

`+ -` menor precedência
`%`
`* /` maior precedência

Incremento e decremento tem precedência maior que operadores aritméticos.

`a*b++`

Equivale a

`(a)*(b++)`

Incremento e decremento só podem ser usados com variáveis. Não com expressões ou constantes.

- `5++;` (errado)

- `(a+b)++;` (errado)

veja o programa:

```
a = 2 ;
b = 5 ;
n = ( a + b++)*3;
```

Executando o programa teremos

$$n = (2 + 5) * 3 = 7 * 3 = 21;$$

Outro exemplo.

```
b = 1;
n = b++ + b++ + b++;
printf("%d\n",n);
printf("%d\n",b);
```

Vai imprimir 3 e 4.

3.8 Operadores Relacionais `> < >= <= == !=`

Operador	Significado
`>`	maior
`>=`	maior igual
`<`	menor
`<=`	menor igual
`==`	igualdade
`!=`	diferente

Veja o programa:

```
#include <stdio.h>
int main(void){

  int verdad, falso;
  verdad = ( 15 < 20 );
  falso = ( 15 == 20 );
  printf("Verdadeiro = %d, falso = %d\n",verdad,falso);
  return 1;
}
```

A saída do programa será:

```
Verdadeiro = 1, falso = 0
```

3.8.1 Precedência

A precedência dos operadores relacionais é mais baixa. Isto significa que serão avaliados depois.

```
3+3==5
```

Equivale a

```
(3+3)==5
```

Referências

[1] Henrique José dos Santos. Curso de linguagem c, ufmg. Universidade Federal de Minas Gerais.

[2] Flávio Keidi Miyazawa. Notas de aula de algoritmos e programação de computadores. Colaboradores : Cid Carvalho de Souza e Tomasz Kowaltowski.

[3] Victorine, Viviane, and Mizrahi. Treinamento em linguagem c, curso completo, módulo 1.