MC102 - Algoritmos e programação de computadores

Aula 4: Variáveis II, Comandos de entrada, atribuiçoes e operações aritméticas

A função `scanf`

- realiza a leitura de um texto a partir do teclado
- parâmetros:
  - uma string, indicando os tipos das variáveis que serão lidas e o formato dessa leitura.
  - uma lista de variáveis
- aguarda que o usuário digite um valor e atribui o valor digitado à variável

A função `scanf`

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

A função `scanf`

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

A função `scanf`

Leitura de várias variáveis

```
#include <stdio.h>
main(){
  int m, n, o;
  printf("Digite três números: ");
  scanf("%d %d %d",&m, &n, &o);
  printf("O valores digitados foram\
          %d %d %d\n", m, n, o);
}
```

O endereço de uma variável

- Toda variável tem um endereço de memória associado a ela. Esse endereço é o local onde essa variável é armazenada no sistema (como se fosse o endereço de uma casa, o local onde as pessoas "são armazenadas").

O endereço de uma variável

- Normalmente, o endereço das 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.
- O endereço de uma mesma variável pode mudar entre diferentes execuções de um mesmo programa C usando uma mesma máquina.

O operador “address-of” & de C

o operador & retorna o endereço de uma determinada variável

Ex: `printf ("%d", &valor);`

imprime o endereço da variável `valor`.

O operador "address-of" & de C

- É necessário usar o operador `&` no comando `scanf`, pois esse operador indica que o valor digitado deve ser colocado no endereço referente a uma variável.
- **Esquecer de colocar o `&` comercial é um erro muito comum que pode ocasionar erros de execução.**

O operador “address-of” & de C

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

```
#include <stdio.h>
int main(void){
  int n = 8;
  printf("valor %d, endereço 0x%x\n",n,&n);
}
```

Formatos de leitura de variável

Os formatos de leitura são muito semelhantes aos formatos de escrita utilizados pelo `printf`. A tabela a seguir mostra alguns formatos possíveis de leitura

Código	Função
%c	Lê um único caracter
%s	Lê uma série de caracteres

Formatos de leitura de variável

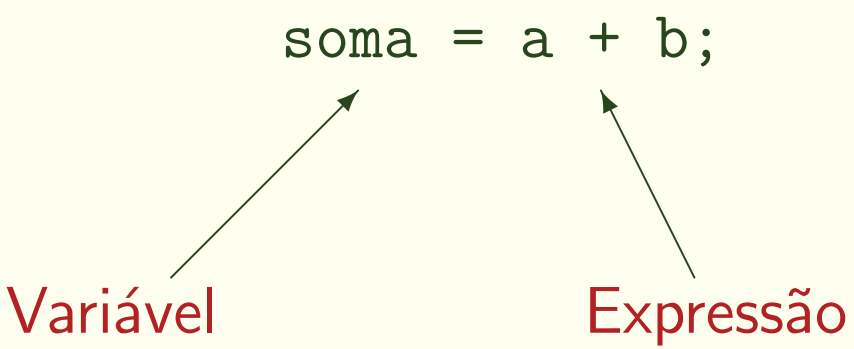
Código	Função
%d	Lê um número decimal
%u	Lê um decimal sem sinal
%l	Lê um inteiro longo
%f	Lê um número em ponto flutuante
%lf	Lê um double

Atribuição

Atribuir um valor de uma expressão a uma variável significa calcular o valor daquela expressão e copiar aquele valor para uma determinada variável.

Atribuição

No exemplo abaixo, a variável `soma` recebe o valor calculado da expressão `a + b`



Atribuição

- O operador de atribuição é o sinal de igual ($=$)

À esquerda do operador de atribuição deve existir somente o nome de uma **variável**.

$=$

À direita, deve haver uma **expressão** cujo valor será calculado e armazenado na variável

Expressão

- Uma expressão é um conjunto de operações aritiméticas, lógicas ou relacionais utilizados para fazer "cálculos" sobre os valores das variáveis.

  Ex: `a + b`

  Calcula a soma de a e b

Expressões

- Uma constante é uma expressão e como tal, pode ser atribuída a uma variável (ou em qualquer outro lugar onde uma expressão seja necessária)

  Ex: `a = 10;`

- Uma variável também é uma expressão

  Ex: `a = b;`

Expressões

- $< expressao > + < expressao >$: Calcula a soma de duas expressões.

  Ex: `a = a + b;`

- $< expressao >$ - $< expressao >$: Calcula a subtração de duas expressões.

  Ex: `a = a - b;`

- $< expressao >$ * $< expressao >$: Calcula o produto de duas expressões.

  Ex: `a = a * b;`

Expressões

- $< expressao > / < expressao >$: Calcula o quociente de duas expressões.

  Ex: `a = a / b;`

- $< expressao >$ % $< expressao >$: Calcula o resto da divisão (inteira) de duas expressões.

  Ex: `a = a % b;`

- - $< expressao >$: Calcula o oposto da expressão.

  Ex: `a = -b;`

Exercício: Qual o valor da expressão

$5 + 10 \% 3$?

E da expressão 5 * 10 % 3?

Precedência

- Precedência é a ordem na qual os operadores serão calculados quando o programa for executado. Em C, os operadores são calculados na seguinte ordem:
  - * e /, na ordem em que aparecerem na expressão.
  - %
  - + e -, na ordem em que aparecerem na expressão.

Alterando a precedência

- $(< expressao >)$ também é uma expressão, que calcula o resultado da expressão dentro dela para só então permitir que as outras expressões executem. Deve ser utilizada quando a ordem da precedência não atende aos requisitos do programa.

  Ex: `5 + 10` % 3 retorna 6, enquanto `(5 + 10)` % 3 retorna 0

- Você pode usar quantos parênteses desejar dentro de uma expressão, contanto que utilize o mesmo número de parênteses para abrir e fechar expressões.

Incremento(++) e Decremento(−−)

- Operadores de incremento e decremento tem duas funções: servem como uma expressão e incrementam ou decrementam o valor da variável ao qual estão associados em uma unidade.

  Ex: c++ — incrementa o valor da variável c em uma unidade

- Dependendo da posição do operador de incremento e decremento, uma função é executada antes da outra.

Incremento(++) e Decremento(−−)

- operador a direita da variável: Primeiro a variável é incrementada, depois a expressão retorna o valor da expressão. Ex:

```
#include <stdio.h>
main () {
  int a = 10;
  printf ("%d", ++a);
}
```

Imprime 11

Incremento(++) e Decremento(−−)

- operador a direita da variável: Primeiro a expressão retorna o valor da variável, e depois a variável é incrementada. Ex:

```
#include <stdio.h>
int main (void) {
  int a = 10;
  printf ("%d", a++);
}
```

Imprime 10

Incremento(++) e Decremento(−−)

- Em uma expressão, os operadores de incremento e decremento são sempre calculados primeiro (tem maior precedência)

```
#include <stdio.h>
int main (void) {
  int a = 10;
  printf ("%d", a * ++a);
}
```

Imprime 121

Atribuições simplificadas

Uma expressão da forma

```
a = a + b
```

onde ocorre uma atribuição a uma das variáveis da expressão pode ser simplificada como

```
a += b
```

Atribuições simplificadas

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;`