ALGORITMOS E ESTRUTURAS DE DADOS I

INTRODUÇÃO AO C

Prof.: Edwaldo Soares Rodrigues

E-mail: edwaldoroadsf1@yahoo.com.br

UFOP departamento de computação

UNIVERSIDADE DO ESTADO DE MINAS GERAIS | UEMG

UNIDADE DIVINÓPOLIS

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`

```
soma = a + b;
```

Variável

Expressão

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;`

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.

Exercício: Qual o valor da expressão

$5 + 10 \% 3$?

E da expressão 5 * 10 % 3?

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;

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`.

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);
}
```

Expressões relacionais

Expressões relacionais são aquelas que realizam uma comparação entre duas expressões e retornam

1. Zero (0), se o resultado é falso
2. Um (1), ou qualquer outro número diferente de zero, se o resultado é verdadeiro.

Expressões relacionais

- $< expressao > == < expressao >$: Retorna verdadeiro quando as expressões forem iguais.

  Ex: `a == b`

- $< expressao >$!= $< expressao >$: Retorna verdadeiro quando as expressões forem diferentes.

  Ex: `a != b`

Expressões relacionais

- $< expressao >$ `>` $< expressao >$: Retorna verdadeiro quando a expressão da esquerda tiver valor maior que a expressão da direita.

  Ex: `a > b`

- $< expressao >$ `<` $< expressao >$: Retorna verdadeiro quando a expressão da esquerda tiver valor menor que a expressão da direita.

  Ex: `a < b`

Expressões relacionais

- $< expressao >$ `>=` $< expressao >$: Retorna verdadeiro quando a expressão da esquerda tiver valor maior ou igual que a expressão da direita.

  Ex: `a >= b`

- $< expressao >$ `<=` $< expressao >$: Retorna verdadeiro quando a expressão da esquerda tiver valor menor ou igual que a expressão da direita.

  Ex: `a <= b`

Expressões básicas

- Toda constante inteira ou caracter é uma expressão.

  Ex: 1

- Toda variável inteira ou caracter também é uma expressão.

  Ex: a

Expressões lógicas

Expressões lógicas são aquelas que realizam uma operação lógica (`ou`, `e`, `não`, etc...) e retorna verdadeiro ou falso (como as expressões relacionais)

Expressões lógicas

- $< expressao >$ `&&` $< expressao >$: Retorna verdadeiro quando ambas as expressões são verdadeiras. Sua tabela verdade é:

Op_1	Op_2	Ret
V	V	V
V	F	F
F	V	F
F	F	F

Ex: `a == 0 && b == 0`

Expressões lógicas

- $< expressao >$ || $< expressao >$: Retorna verdadeiro quando pelo menos uma das expressões é verdadeiras. Sua tabela verdade é:

Op_1	Op_2	Ret
V	V	V
V	F	V
F	V	V
F	F	F

Ex: `a == 0 || b == 0`

Expressões lógicas

- ! $< expressao >$: Retorna verdadeiro quando a expressão é falsa e vice-versa. Sua tabela verdade é:

Op_1	Ret
V	F
F	V

Ex: `!(a == 0)`

Simplificações úteis

- `!(a == b)` é equivalente a `a != b`
- `!(a != b)` é equivalente a `a == b`
- `!(a > b)` é equivalente a `a <= b`
- `!(a < b)` é equivalente a `a >= b`
- `!(a >= b)` é equivalente a `a < b`
- `!(a <= b)` é equivalente a `a > b`

Comandos condicionais

Um comando condicional é aquele que permite decidir se um determinado bloco de comandos deve ou não ser executado, a partir do resultado de uma expressão lógica.

Comandos condicionais

- O principal comando condicional da linguagem C é o `if`, cuja sintaxe é:

  ```
  if (expressão lógica)
      comando;  ou
  if (expressão lógica) {
      comandos
  }
  ```

- Os comandos são executados somente se a expressão lógica for verdadeira.

Comandos condicionais

- Uma variação do comando `if` é o `if/else`, cuja sintaxe é:

```
if (expressão lógica) {
    comandos executados se a expressão é verdadeira
} else {
    comandos executados se a expressão é falsa
}
```

Decisão múltipla

- Dependendo do problema proposto, o programa pode ser formado por um conjunto muito grande de comandos `if` e expressões lógicas.

  Ex: Faça um programa que, dado um RA, emite uma mensagem se o aluno estiver matriculado em uma turma de MC102.

Decisão simples

Para apenas um aluno, a solução seria:

```
main () {
   int a;
   scanf("%d", &a);
   if (a == 10129) {
      printf("O aluno %d está matriculado\n", a);
   }
}
```

Decisão múltipla

Para dois alunos, a solução seria:

```
main () {
  int a;
  scanf("%d", &a);
  if (a == 10129 a == 16267) {
      printf("O aluno %d está matriculado\n", a);
  }
}
```

Decisão múltipla

- Problema: cada turma de MC102 possui cerca de 60 alunos e temos 14 turmas neste semestre.

```
if (a == 2582 || a == 10129 ||
    a == 16267 || ...
    a = 962185) {
    printf("O aluno %d está matriculado\n", a);
}
```

- Teríamos muitas condições a serem testadas.

Decisão múltipla

- Faça um programa que, dado um RA, mostre o nome desse aluno.

Decisão simples

Para apenas um aluno, a solução seria:

```
main () {
  int a;
  scanf("%d", &a);
  if (a == 10129) {
     printf("Maria Cândida Moreira Telles\n", a);
  }
}
```

Decisão múltipla

```
main () {
  int a;
  scanf("%d", &a);
  if (a == 10129)
     printf("Maria Cândida Moreira Telles\n");
  if (a == 33860)
     printf("Larissa Garcia Alfonsi\n");
  if (a == 33967)
     printf("Leonardo Kozlowiski Kenupp\n");
}
```

Decisão múltipla

- Novamente, temos um conjunto muito grande de alunos.
- Além disso, não podemos utilizar os operadores lógicos que utilizamos anteriormente.
- Podemos tentar diminuir o número de testes realizados?

Decisão múltipla

```
main () {
  int a;
  scanf("%d", &a);
  if (a == 10129)
     printf("Maria Cândida Moreira Telles\n");
  else if (a == 33860)
     printf("Larissa Garcia Alfonsi\n");
  else if (a == 33967)
     printf("Leonardo Kozlowiski Kenupp\n");
}
```

O comando `switch`

- O objetivo do comando `switch` é simplificar uma expressão onde uma variável inteira ou caracter deve fazer diferentes operações dependendo exclusivamente de seu valor. Sua sintaxe é:

```
switch (variável inteira) {
    case valor:  comandos
    break;
    case valor:  comandos
    break;
}
```

O comando `switch`

```
switch(a) {
case 10129:
    printf("Maria Cândida Moreira Telles\n");
    break;
case 33860:
    printf("Larissa Garcia Alfonsi\n");
    break;
case 33967:
    printf("Leonardo Kozlowiski Kenupp\n");
    break;
}
```

O comando `switch`

- Os comandos começam a ser executados a partir do ponto onde o valor da variável corresponde ao valor antes dos dois pontos (:).
- Executa todos os comandos até que encontre um comando `break` ou que chegue ao final do bloco de comandos do `switch`

Valor padrão

- Você pode utilizar, ao invés de um valor, o valor `default`. A execução dos comandos inicia no comando `default` se nenhum outro valor for correspondente ao valor da variável. Sua sintaxe é:

```
switch (variável inteira) {
    valor:   comandos break;
    default:   comandos
}
```

Valor padrão

```
switch(a) {
case 10129:
    printf("Maria Cândida Moreira Telles\n");
    break;
case 33860:
    printf("Larissa Garcia Alfonsi\n");
    break;
default:
    printf("O aluno não está matriculado\n");
}
```