INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SUL-RIO-GRANDENSE
Campus Passo Fundo

ALGORITMOS II

Prof. Adilso Nunes de Souza

ROTEIRO

- Escrita formatada
- Funções matemáticas
  - Pow
  - Sqrt
  - Fabs
  - Ceil
  - Floor
  - Round

FORMATAÇÃO

- A função printf fornece aos usuários de C múltiplas maneiras de formatar a exibição dos dados na tela do computador, conforme já estudado.
  - printf(“expressão de controle”,argumentos);
    - Ex: printf("Resultado: %.2f", 1.45267);
- A classe cout também fornece inúmeras facilidades para os usuários de C++.

NOTAÇÃO

- Podemos escolher a notação utilizada para exibição de números reais.

```
cout << fixed;
```

- Instruímos o programa a exibir valores reais usando a notação de ponto fixo.

```
cout << scientific;
```

- Instruímos o programa a utilizar a notação científica.

NOTAÇÃO

- Exemplos:

```
cout << fixed;
cout << 12.4576;
```

Exibe: 12.4576

```
cout << scientific;
cout << 12.4576;
```

Exibe: 1.245700^{e+001}

PRECISÃO

- O segundo método é utilizado para fixar a precisão de variáveis reais, ou seja, o número mínimo de casas decimais à serem exibidas após a vírgula em um valor real:

  cout.precision (valor);

- Onde valor representa o número de casas.
- Por default, C++ utiliza 6 casas decimais após a vírgula. Quando alteramos o valor da precisão, este novo valor vale para todas as utilizações futuras de cout;

PRECISÃO

- Para surtir efeito é necessário atribuir a notação fixed para exibição dos valores;

```
cout << fixed;
cout.precision(2);
cout << 12.4576;
```

Exibe: 12.46

- O valor obedece a regra de arredondamento, se menor que 5 para baixo, caso contrário para cima.

ESPAÇAMENTO

- O comando cout permite também escolher um número mínimo de caracteres para ser exibido na tela. Isto é feito utilizando o método:

  cout.width(x);

- Onde substituímos x pelo número mínimo de caracteres a ser exibido na tela. Após a utilização deste método, utilizamos o comando cout para exibir o valor desejado.

ESPAÇAMENTO

- Exemplo:

```
int variavel = 10;
cout.width ( 5 );
cout << variavel;
```

- Neste exemplo, foi especificado cout.width (5); e o valor a ser exibido é 10.
- Assim, cout predecerá o valor 10 com três espaços em branco.

ESPAÇAMENTO

- Observe que o valor especifica o número mínimo de caracteres que a saída consumirá. Se o valor a ser exibido requer mais caracteres do que o especificado, será usado o número de caracteres necessários para exibir o valor corretamente.
- É importante observar também que o método cout.width só é válido para a próxima utilização de cout: após isto, o número mínimo de caracteres volta a ser zero.

ESPAÇAMENTO

- Podemos também determinar o caractere a ser utilizado para preencher os espaços em branco de um campo de exibição. Isto é feito com o seguinte método:

  cout.fill (‘caractere’);

- Onde substituímos “caractere” pelo caractere que será exibido. É necessário utilizar aspas entre o caractere.

ESPAÇAMENTO

- O exemplo abaixo mostra a utilização conjunta destes dois métodos:

```
int variavel = 10;
cout.width ( 8 );
cout.fill('0');
cout << variavel;
```

- Este exemplo fará a seguinte exibição na tela:

```
00000010
```

ALINHAMENTO

- É possível definir juntamente com o comando width qual o alinhamento do conteúdo, por padrão o preenchimento é sempre à esquerda e o conteúdo à direita, mas poderá ser alterado:
- Left
- Right
- Internal

ALINHAMENTO

- left: o conteúdo fica à esquerda e o preenchimento é incluído à direita.
- Exemplo:

```
v = -48.6;
cout.width(10);
cout.fill('*');
 cout << left << v << endl;
//saída: -48.6*****
```

ALINHAMENTO

- right: o conteúdo fica à direita e o preenchimento é incluído à esquerda.
- Exemplo:

```
v = -48.6;
cout.width(10);
cout.fill('*');
 cout << right << v << endl;
//saída: *****-48.6
```

ALINHAMENTO

- internal: a saída é preenchida para a largura do campo inserindo caracteres de preenchimento entre o sinal de pontuação e o valor.
- Exemplo:

```
v = -48.6;
cout.width(10);
cout.fill('*');
 cout << internal << v << endl;
//saída: -*****48.6
```

ALINHAMENTO

- A função width é membro da biblioteca padrão iostream, mas existem outra biblioteca que dispõem de opções semelhantes, no caso a biblioteca iomanip
- A função setw permite especificar a largura de elementos de dados na mesma linha
- Assim como a função width só afeta a primeira saída após o comando.

ALINHAMENTO

- Exemplo:

```
v = -48.6;
cout << setw(10)  << v;
//saída:      -48.6
```

- Irá incluir antes da saída os espaços em branco necessários para completer o tamanho 10, usando o setfill é possível definir o caracter de preenchimento:

```
cout << setfill('-');
cout << setw(10) << left << v;
//saída: -48.6-----
```

POW

- A biblioteca cmath é usada para execução de várias operações matemáticas, entre as principais destaca-se:
  - pow: calcula a potência de um número elevado a outro.
  - Sintaxe: pow(x,y);
  - Ex: pow(3,2);

  Retorna 9.

SQRT

- sqrt: usado para calcular a raíz quadrada de um determinado valor;
  - Sintaxe: sqrt(x);
  - Ex: sqrt(4);

  Retorna 2;

FABS

- fabs: obtém o valor absoluto de um número, sem considerar o sinal;
  - Sintaxe: fabs(x);
  - Ex: fabs(-8);

  Retorna 8.

CEIL

- ceil: arredonda um número real para cima;
  - Sintaxe: ceil(num);
  - Ex: ceil(3.6)

  Retorna 4.

FLOOR

- floor: arredonda um número real para baixo.
  - Sintaxe: floor(num)
  - Ex: floor(5.7)

  Retorna 5.

ROUND

- round: comando usado para arredondar um número float tendo como critério o primeiro valor decimal. Se menor que 5 arredonda para baixo, se igual ou maior que 5 arredonda para cima, retornando um número inteiro.
- Sintaxe: round(num)
- Ex: round(5.49)

  Retorna 5

  round(5.632)

  Retorna 6

ORDEM DE PRECEDÊNCIA

- O cálculo de expressões é um processo complexo que necessita levar em consideração a relação de precedência entre os operadores envolvidos.
- Exemplo:

  A + B * C → A + (B * C)

  A % B - ++C → ((A % B) – (++C))

  A + (B < C) → Operadores relacionais tem maior prioridade

ORDEM DE PRECEDÊNCIA

- Tabela dos principais operadores:

OPERADORES	ASSOCIATIVIDADE
() .	→
++ -- ! & * (*casting*) sizeof	←
* / %	→
+ -	→
< <= >= >	→
== !=	→
^	→
&&	→
||	→
= *op*=	←

ORDEM DE PRECEDÊNCIA

- A associatividade diz respeito à forma como a expressão é avaliada.
- Por exemplo:

  a + b + c

- É avaliada como se tivesse sido escrita

  (a + b) + c

- Pois os operadores aditivos (e aritméticos em geral) são associativos da esquerda para a direita.

ORDEM DE PRECEDÊNCIA

- Já o operador de atribuição = é associativo da direita para a esquerda, de modo que a expressão:

  a = b = 10

- É avaliada como se tivesse sido escrita:

  a = (b = 10)

- Ou seja, como se fossem duas instruções separadas e sequenciais sendo executadas, primeiro b recebe 10 e a recebe o valor de b, no caso 10.

ORDEM DE PRECEDÊNCIA

- Analise este cenário e resolva a expressão:

```
b = 10;
a = 3;
c = a * 3 + (b + 10) / 2
```

ORDEM DE PRECEDÊNCIA

- Resolvendo:

```
b = 10;
a = 3;
c = a * 3 + (b + 10) / 2
```

- o valor da variável a é multiplicado por 3, resultando em 9
- o valor de b somado a 10, resultando em 20 (parênteses em ação)
- o resultado desta soma é dividido por 2, resultando em 10 (ordem de precedência de operadores em c++)
- o resultado da multiplicação de a com as operações sobre b é a soma desses valores, resultando em 19
- e então o resultado da expressão é atribuído à variável c.

REFERÊNCIAS

- SCHILDT Herbert. C Completo e Total 3ª edição
- KERNIGHAN Brian W. C a linguagem de programação
- ASCENCIO, Ana Fernanda Gomes. CAMPOS, Edilene Aparecida Veneruchi de. *Fundamentos da programação de computadores*. 2 ed. São Paulo: Pearson, 2007.