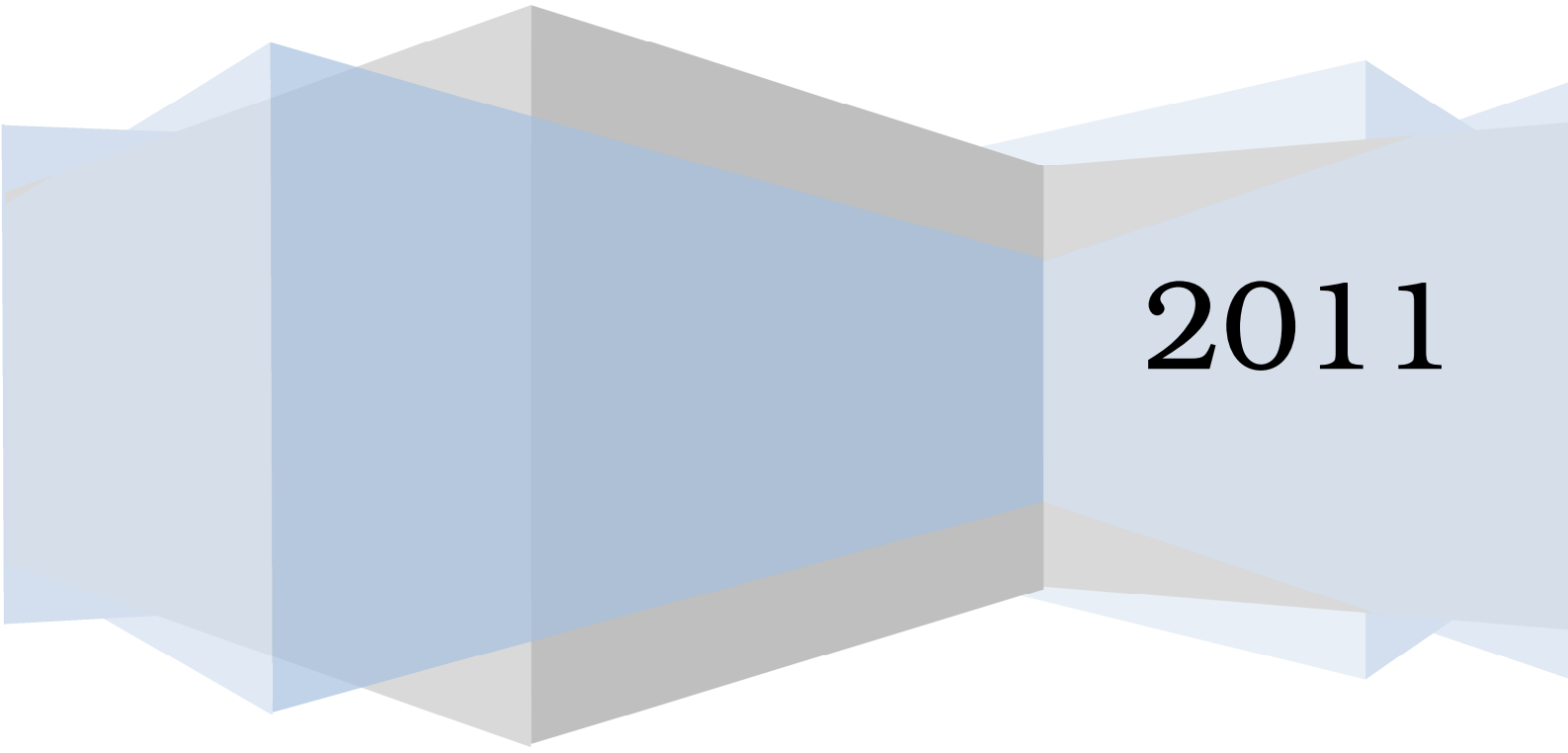
Flávio Augusto de Freitas
Introdução à Programação em Linguagem C/C++

http://flavioaf.blogspot.com

C/C++

Tutorial 2 (usando Dev-C++ versão 4.9.9.2)



2011

1 Introdução

Esta série de tutoriais foi escrita usando o **Microsoft Windows 7 Ultimate** e o **Bloodshed Dev-C++** versão 4.9.9.2, que pode ser baixada em http://www.bloodshed.net.

2 Escrevendo Informações

Execute o Dev-C++. Depois de carregado o programa, clique no menu Arquivo, depois aponte o mouse para a opção Novo e clique em Projeto... . Na janela que se abre, clique em

Ainda na mesma janela, onde está escrito Projeto1, mude para **Escrevendo**. Clique no botão Ok. Agora, vamos salvar previamente nosso projeto. Na janela que se abriu, no lado esquerdo, clique no botão Área de Trabalho (ou Desktop, se estiver escrito).

Clique no botão  para criar uma nova pasta na Área de Trabalho. Digite o nome `ProjetoEscrevendo` e pressione a tecla Enter. Dê dois cliques na pasta ProjetoEscrevendo, recém criada. Em seguida, na mesma janela, observe que o projeto já tem nome (`Escrevendo.dev`). Clique no botão Salvar.

Modifique o código do programa para que fique como abaixo:

```
#include <cstdlib>
#include <iostream>

using namespace std;

int main(int argc, char *argv[])
{
  printf("\n%2d",350);
  printf("\n%4d",350);
  printf("\n%6d\n\n",350);

  printf("\n%4.2f",3456.78);
  printf("\n%3.2f",3456.78);
  printf("\n%3.1f",3456.78);
  printf("\n%10.3f\n\n",3456.78);

  printf("\n%10.2f %10.2f %10.2f",8.0,15.3,584.13);
  printf("\n%10.2f %10.2f %10.2f\n\n",834.0,1500.55,4890.21);

  printf("\n%04d",21);
  printf("\n%06d",21);
  printf("\n%6.4d",21);
  printf("\n%6.0d\n\n",21);

  printf("%d %c %x %o\n",'A','A','A','A');
  printf("%c %c %c %c\n\n",'A',65,0x41,0101);

  system("PAUSE");
  return EXIT_SUCCESS;
}
```

Após a digitação, pressione Ctrl-S para salvar o arquivo. Pressione F9 para executar o programa. Veja a saída do programa e compare com o código que você escreveu. Tente fazer a analogia entre o que você codificou e o que o programa mostrou na tela. Pressione qualquer tecla para voltar ao compilador.

3 Lendo e Escrevendo

Muitas vezes um programa precisa obter alguma informação do mundo exterior. Isto é feito através de comandos de leitura. Vamos criar um novo projeto.

Clique no menu Arquivo, depois aponte o mouse para a opção Novo e clique em Projeto... . Na janela que se abre, clique em

Ainda na mesma janela, onde está escrito Projeto1, mude para **LendoEscrevendo**. Clique no botão Ok. Agora, vamos salvar previamente nosso projeto. Na janela que se abriu, no lado esquerdo, clique no botão Área de Trabalho (ou Desktop, se estiver escrito).

Clique no botão  para criar uma nova pasta na Área de Trabalho. Digite o nome `ProjetoLendoEscrevendo` e pressione a tecla Enter. Dê dois cliques na pasta ProjetoLendoEscrevendo, recém criada. Em seguida, na mesma janela, observe que o projeto já tem nome (`LendoEscrevendo.dev`). Clique no botão Salvar.

Modifique o código do programa para que fique como abaixo:

```
#include <cstdlib>
#include <iostream>

using namespace std;

int main(int argc, char *argv[])
{
  int num;

  printf("Digite um número: ");
  scanf("%d",&num);
  printf("\no número é %d",num);
  printf("\no endereço e %u",&num);

  system("PAUSE");
  return EXIT_SUCCESS;
}
```

Após a digitação, pressione Ctrl-S para salvar o arquivo. Pressione F9 para executar o programa. Veja a saída do programa e compare com o código que você escreveu. Tente fazer a analogia entre o que você codificou e o que o programa mostrou na tela. Pressione qualquer tecla para voltar ao compilador.

scanf é o comando de leitura mais comum. Para usá-lo, precisamos de uma string de formatação e uma lista de variáveis separadas por vírgula, e cada variável deve ser precedida do símbolo &, que 'mostra' à função scanf para qual endereço de memória a variável está apontando.

Já a função printf foi usada de duas formas, a primeira mostrando o valor da variável num, ou seja, que valor estava armazenado nela. Na segunda forma, foi usado o símbolo & antes da variável, que, como sabemos retorna o endereço da variável.

4 Façamos outro projeto:

Clique no menu Arquivo, depois aponte o mouse para a opção Novo e clique em Projeto... . Na janela que se abre, clique em

Ainda na mesma janela, onde está escrito Projeto1, mude para **LendoEscrevendoCaracteres**. Clique no botão Ok. Agora, vamos salvar previamente nosso projeto. Na janela que se abriu, no lado esquerdo, clique no botão Área de Trabalho (ou Desktop, se estiver escrito).

Clique no botão  para criar uma nova pasta na Área de Trabalho. Digite o nome `ProjetoLendoEscrevendoCaracteres` e pressione a tecla Enter. Dê dois cliques na pasta ProjetoLendoEscrevendoCaracteres, recém criada. Em seguida, na mesma janela, observe que o projeto já tem nome (`LendoEscrevendoCaracteres.dev`). Clique no botão Salvar.

Observação Cada projeto novo deve ficar dentro de uma pasta nova, senão os códigos ficam misturados e você terá problemas com seu programa.

Modifique o código do programa para que fique como abaixo:

```
#include <cstdlib>
#include <iostream>

using namespace std;

int main(int argc, char *argv[])
{
  char ch;
  printf("Pressione uma tecla: ");
  ch=getchar();
  printf("%c\n",ch);

  system("PAUSE");
  return EXIT_SUCCESS;
}
```

Após a digitação, pressione Ctrl-S para salvar o arquivo. Pressione F9 para executar o programa. Veja a saída do programa e compare com o código que você escreveu. Tente fazer a analogia entre o que você codificou e o que o programa mostrou na tela. Pressione qualquer tecla para voltar ao compilador.

5 Strings de Formatação

A documentação mais técnica os chama de "especificadores de conversão", pois o que ocorre na maioria das vezes é, de fato, a conversão de um valor numérico em uma sequência de caracteres que representa aquele valor. Mas o nome "formato" não deixa de estar correto, pois eles especificam em que formato (inteiro, real etc.) está o argumento correspondente.

Código	Conversão/Formato do argumento
%d	Número decimal inteiro (int). Também pode ser usado %i como equivalente a %d.
%u	Número decimal natural (unsigned int), ou seja, sem sinal.
%o	Número inteiro representado na base octal. Exemplo: 41367 (corresponde ao decimal 17143).
%x	Número inteiro representado na base hexadecimal. Exemplo: 42f7 (corresponde ao decimal 17143). Se usarmos %X, as letras serão maiúsculas: 42F7.
%f	Número decimal de ponto flutuante (float). Se quisermos usar um número do tipo double, usamos %lf em vez de %f.
%e	Número em notação científica, por exemplo, 5.97e-12. Podemos usar %E para exibir o E maiúsculo (5.97E-12). 5.97e-12 ⇔ $5.97x10^{-12}$ ⇔ 0,00000000000597.
%g	Escolhe automaticamente o mais apropriado entre %f e %e. Novamente, podemos usar %G para escolher entre %f e %E.
%p	Ponteiro: exibe o endereço de memória do ponteiro em notação hexadecimal.
%c	Caractere: imprime o caractere que tem o código ASCII correspondente ao valor dado.
%s	Sequência de caracteres (string, em inglês).

Observação Se você quiser imprimir um sinal de porcentagem, use %%.

Por exemplo, crie um novo projeto e modifique o código para que fique igual ao seguinte:

```
#include <cstdlib>
#include <iostream>

using namespace std;

int main(int argc, char *argv[])
{
  printf("O lucro do mês foi de 20%%.");

  system("PAUSE");
  return EXIT_SUCCESS;
}
```

Numa sequência de controle, é possível também indicar a largura do campo, o número de casas decimais, o tamanho da variável e algumas opções adicionais. O formato geral é:

```
%[opções][largura][.precisão][tamanho]tipo de dado
```

A única parte obrigatória é o tipo de dado. Todas as outras podem ser omitidas.

Opções

As opções são parâmetros opcionais que alteram a formatação. Você pode especificar zero ou mais delas, colocando-as logo após o sinal de porcentagem:

0	(zero) o tamanho do campo deve ser preenchido com zeros à esquerda quando necessário, se o parâmetro correspondente for numérico.
-	(hífen) o valor resultante deve ser alinhado à esquerda dentro do campo (o padrão é alinhar à direita).
	(espaço) no caso de formatos que admitem sinal negativo e positivo, deixa um espaço em branco à esquerda de números positivos.
+	(adição) o sinal do número será sempre mostrado, mesmo que seja positivo.
'	(apóstrofo) números decimais devem ser exibidos com separador de milhares caso as configurações regionais o especifiquem. Essa opção normalmente só funciona nos sistemas Unix.

Largura do campo

Como o próprio nome já diz, especifica qual a largura mínima do campo. Se o valor não ocupar

toda a largura do campo, este será preenchido com espaços ou zeros. Por exemplo, podemos imprimir um código de até 5 dígitos preenchido com zeros, de maneira que os valores 1, 27, 409 e 55192 apareçam como 00001, 00027, 00409 e 55192.

A largura deve ser especificada logo após as opções, se presentes, e pode ser um número — que especifica a largura — ou um asterisco, que diz que a largura será especificada pelo próximo argumento (ou seja, o argumento anterior ao valor a ser impresso).

Por exemplo, crie um novo projeto e modifique o código para que fique igual ao seguinte, onde o campo terá largura igual ao valor de num (10) e o valor impresso será 300:

```
#include <cstdlib>
#include <iostream>

using namespace std;

int main(int argc, char *argv[])
{
  int num = 10;
  printf("%*d", num, 300);

  system("PAUSE");
  return EXIT_SUCCESS;
}
```

No programa acima, o campo é impresso de acordo com as seguintes regras:

- Se o valor for mais largo que o campo, este será expandido para poder conter o valor. O valor nunca será cortado.
- Se o valor for menor que o campo, a largura do campo será preenchida com espaços ou zeros. Os zeros são especificados pela opção 0, que precede a largura.
- O alinhamento padrão é à direita. Para se alinhar um número à esquerda usa-se a opção - (hífen ou sinal de menos) antes da largura do campo.

Por exemplo, crie um novo projeto e modifique o código para que fique igual ao seguinte e compare as três maneiras de exibir o número 15:

```
#include <cstdlib>
#include <iostream>

using namespace std;

int main(int argc, char *argv[])
{
  printf("%5d", 15);   // exibe "   15"
  printf("%05d", 15);  // exibe "00015"
  printf("%-5d", 15);  // exibe "15   "

  system("PAUSE");
  return EXIT_SUCCESS;
}
```

Para o próximo exemplo, crie um novo projeto e modifique o código para que fique igual ao mostrado abaixo e compare as três maneiras de exibir o número a string “José”:

```
#include <cstdlib>
#include <iostream>

using namespace std;

int main(int argc, char *argv[])
{
  printf("%-10s", "José"); // "José      "
  printf("%10s", "José");  // "      José"
  printf("%3s", "José");   // "José"

  system("PAUSE");
  return EXIT_SUCCESS;
}
```

Precisão

A precisão pode ter quatro significados diferentes:

1. **Se a conversão solicitada for inteira (d, i, o, u, x, X)**: o número mínimo de dígitos a exibir (será preenchido com zeros se necessário).
2. **Se a conversão for real (a, A, e, E, f, F)**: o número de casas decimais a exibir. O valor será arredondado se a precisão especificada no formato for menor que a do argumento.
3. **Se a conversão for em notação científica (g, G)**: o número de algarismos significativos. O valor será arredondado se o número de algarismos significativos pedido for maior que o do argumento.

4. **Se a conversão for de uma sequência de caracteres (s)**: o número máximo de caracteres a exibir.

Assim como a largura do campo, a precisão pode ser especificada diretamente por um número ou com um asterisco, mas deve ser precedida por um ponto.

Para ver o exemplo abaixo, crie um novo projeto e modifique o código para que fique igual ao seguinte:

```
#include <cstdlib>
#include <iostream>

using namespace std;

int main(int argc, char *argv[])
{
  printf("%.5d", 314);       // exibe "00314"
  printf("%.5f", 2.4);       // exibe "2.40000"
  printf("%.5g", 23456789012345);  // "2.3457e+13"
  printf("%.5s", "Bom dia"); // exibe "Bom d"

  system("PAUSE");
  return EXIT_SUCCESS;
}
```

É claro que podemos combinar a largura com a precisão. Por exemplo, %10.4f indica um campo de número real de comprimento total dez e com 4 casas decimais. Note que, na largura do campo, o valor inteiro é levado em conta, inclusive o ponto decimal, e não apenas a parte inteira. Por exemplo, essa formatação aplicada ao número 3.45 irá resultar nisto:

```
"    3.4500"
```

Tamanho da variável

É importante ressaltar que quando são usados modificadores de tamanho de tipos, a maneira como os dados são armazenados pode tornar-se diferente. Assim, devemos informar à função printf() precisamente qual o tipo da variável cujo valor desejamos exibir. A função printf() admite cinco principais modificadores de tamanho de variável:

hh indica que a conversão inteira corresponde a uma variável char. Por exemplo, poderíamos usar o formato %hhd para exibir uma variável do tipo char na base decimal.

h indica que a conversão inteira corresponde a uma variável short.

l indica que a conversão inteira corresponde a uma variável long.

ll indica que a conversão inteira corresponde a uma variável long long.

L indica que a conversão de número real corresponde a uma variável long double.

Quando o tipo da variável não tem modificadores de tamanho (long ou short), não se usa nenhum modificador de tamanho da variável na função printf().

Sequências de escape

Sequências de escape são combinações de caracteres que têm significado especial, e são sempre iniciadas por uma barra invertida (\). Você pode usá-las em qualquer literal de caractere ou string. Por exemplo, a string "`linha 1\nlinha 2`" equivale a:

```
linha 1

linha 2
```

pois a sequência \n indica uma quebra de linha. A seguir apresentamos a tabela com as sequências de escape suportadas pela linguagem C:

Sequência	**Significado**
\n	Quebra de linha (line feed ou LF)
\t	Tabulação horizontal
\b	Retrocede o cursor em um caractere (backspace)
\r	Retorno de carro (carriage return ou CR): volta o cursor para o começo da linha sem mudar de linha
\a	Emite um sinal sonoro
\f	Alimentação de formulário (form feed ou FF)
\v	Tabulação vertical (em impressoras)
\"	Aspa dupla
\'	Aspa simples
\\	Barra invertida
\0	Caractere nulo (caractere de valor zero, usado como terminador de strings)

\N	O caractere cuja representação octal é N (dígitos de 0 a 7)
\xN	O caractere cuja representação hexadecimal é N (dígitos de 0 a 9 e de A a F)

6 Saindo do Dev-C++

Agora que terminamos este programa, podemos sair do ambiente do Dev-C++. Para isto, clique no menu Arquivo e depois clique na opção Sair. Caso alguma coisa não esteja salva, o Dev-C++ mostrará uma tela de confirmação.

Basta clicar no botão Yes para o Dev-C++ salvar o arquivo e terminar. Se desistir de sair do Dev-C++, basta clicar no botão Cancel e você continuará com o Dev-C++ aberto.

7 Terminamos

Terminamos por aqui.

Corra para o próximo tutorial.