

# Estruturas de Dados

Prof. Rodrigo Martins

[rodrigo.martins@francomontoro.com.br](mailto:rodrigo.martins@francomontoro.com.br)

# Agenda

- Apresentação da disciplina
- Objetivos
- Conteúdo Programático
- Metodologia de Trabalho
- Ferramenta de Apoio
- Nivelamento de Algoritmos (Revisão)

# Ementa

- Armazenamento e recuperação aleatória: vetores, matrizes, registros, alocação dinâmica, ordenação para otimização de busca, algoritmos (busca, inserção, remoção e alteração).
- Armazenamento e recuperação na mesma ordem: fila, implementação, algoritmos.
- Armazenamento e recuperação em ordem inversa: pilha, implementação, algoritmos.
- Noções de memória secundária: acesso sequencial e acesso aleatório.

# Objetivos

- Apresentar ao aluno o conceito de abstração de dados, sua importância para os princípios de modularidade, encapsulamento e independência de implementação.
- Apresentar as estruturas de dados clássicas, suas características funcionais, formas de representação, operações associadas e complexidade das operações.
- Ao final da disciplina, o aluno estará capacitado a selecionar as estruturas de dados, os algoritmos de classificação de dados e as respectivas representações que sejam mais adequadas a uma dada aplicação, implementando-as com uso dos recursos de linguagem de programação mais apropriados ao caso.

# Conteúdo Programático

- Módulo 1
  - Algoritmos e a Solução de Problemas
  - Nivelamento de algoritmos
- Módulo 2
  - Funções
  - Módulos
  - Escopo de Variável
  - Vetores ou Arrays
  - Matrizes ou Arrays Multidimensionais

# Conteúdo Programático

- Módulo 3
  - Structs
  - Ponteiros
  - Ponteiros de Structs
- Módulo 4
  - Estruturas de Dados Avançadas
  - Pilhas

# Conteúdo Programático

- Módulo 5
  - Estruturas de Dados Avançadas
  - Filas
- Módulo 6
  - Listas

# Conteúdo Programático

- Módulo 7
  - Ordenação
    - Importância da Ordenação
    - Eficiência
  - Tipos de Ordenação
    - Bubble Sort
    - Selection Sort
    - Quick Sort
    - Merge Sort



# Critérios de Avaliação

- T1 – Lista de Exercícios (30%)
  - P1 - Avaliação Bimestral (70%)
- } MB1
- T2 - Lista de Exercícios (30%)
  - P2 – Avaliação Bimestral (70%)
- } MB2
- Média Final –  $(MB1 + MB2) / 2$
- Média final maior ou igual a 7,0 (sete) implicará em aprovação sem exame final;
  - Média final igual ou superior a 4,0 (quatro) e inferior a 7,0 (sete) dependerá de aprovação em exame final;
  - Média final de aproveitamento inferior a 4,0 (quatro) implicará em reprovação;
  - A aprovação em exame final será obtida se a média aritmética da média final de aproveitamento com a nota do exame final for igual ou superior a 5,0 (cinco).

# Metodologia de Trabalho

- Material exposto em sala de aula (Apresentações);
- Indicação de Sites sobre o conteúdo (Artigos);
- Exemplos e Exercícios práticos;
- Uso de metodologias ativas para desenvolvimento de projetos.

# Bibliografia Básica

- VELOSO, P. et Alli. Estruturas de Dados. Ed. Campus, 1986.
- TANENBAUM, A., LANGSAM, Y., AUGUSTEIN, M. Estruturas de Dados Usando C. Makron Books, 1995.
- PEREIRA, Silvio do Lago. Estrutura de Dados Fundamentais: conceitos e aplicações. Editora Erica. São Paulo. 1996.

# Bibliografia Complementar

- VILLAS, Marcos Vianna. Estrutura de Dados: conceitos e técnicas de implementação. Editora Campus. Rio de Janeiro. 1993.
- SAVARCFILTER, Jayme Luiz. Estrutura de Dados e seus Algoritmos. LTC. Rio de Janeiro. 1994.
- WIRTH, NIKLAUS. Algoritmos e Estruturas de Dados. Prentice-Hall do Brasil, Rio de Janeiro, 1999.

# Ferramenta de Apoio

- CodeBlocks
- <https://sourceforge.net/projects/codeblocks/files/Binaries/20.03/Windows/codeblocks-20.03mingw-setup.exe>

# Programas C++

- Essencialmente, um programa C++ consiste de uma ou mais partes chamadas funções. Além disso, um programa em C++ deve definir pelo menos uma função chamada main. Esta função marca o ponto de início de execução do programa. A linguagem C++ é case sensitive. Programas C++ tem a seguinte estrutura geral:

```
#include <iostream>
using namespace std;

definição de constantes

funções

int main()
{
    declaração de variáveis
    ....
    sentenças
    ....
}
```

# Variáveis em C++

- Os tipos básicos de dados existentes em C++ são:

Tipo de Dado	Bits	Faixa de Valores
<b>char</b>	8	-128 a 127
<b>bool</b>	8	true ou false
<b>int</b>	32	-2.147.483.647 a 2.147.483.647
<b>float</b>	32	7 dígitos significativos
<b>double</b>	64	15 dígitos significativos

# Variáveis em C++

- Abaixo está um exemplo de um programa com diversas definições de variáveis:

```
int main()
{
    int pera;
    char qualidade;
    float peso;

    pera = 3;
    qualidade = 'A';
    peso = 0.653;
    ...
}
```



# Variáveis em C++

Para resumir: quando um programa é executado, uma variável é associada com:

- **um tipo:** diz quantos bytes a variável ocupa, e como ela deve ser interpretada.
- **um nome:** um identificador.
- **um endereço:** o endereço do byte menos significativo do local da memória associado a variável.
- **um valor:** o conteúdo real dos bytes associados com a variável; o valor da variável depende do tipo da variável; a definição da variável não dá valor a variável; o valor é dado pelo operador de atribuição, ou usando a função cin.

# Constantes em C++

- Pode-se usar também números ou caracteres cujos valores não mudam. Eles são chamados de **constantes**.
- Assim como variáveis, constantes também têm tipos. Uma constante pode ser do tipo int, char, etc.
- Não há a necessidade de declarar constantes, e pode utilizá-las diretamente (o compilador reconhece o tipo pela maneira que são escritos).
- Por exemplo, 2 é do tipo int, e 2.0 é do tipo double. Por convenção, todas as constantes reais são do tipo double.
- Uma constante caractere é escrita entre apóstrofes, como em 'A'.

```
#define PRECO 1.99
```

# Entrada e Saída

- Se quisermos que um programa C++ mostre alguns resultados, ou se quisermos que o programa peça ao usuário que entre com alguma informação, usa-se os elementos **cout** e **cin**, para isso deve-se incluir as seguintes linhas no início do seu código fonte:

```
#include <iostream>  
using namespace std;
```

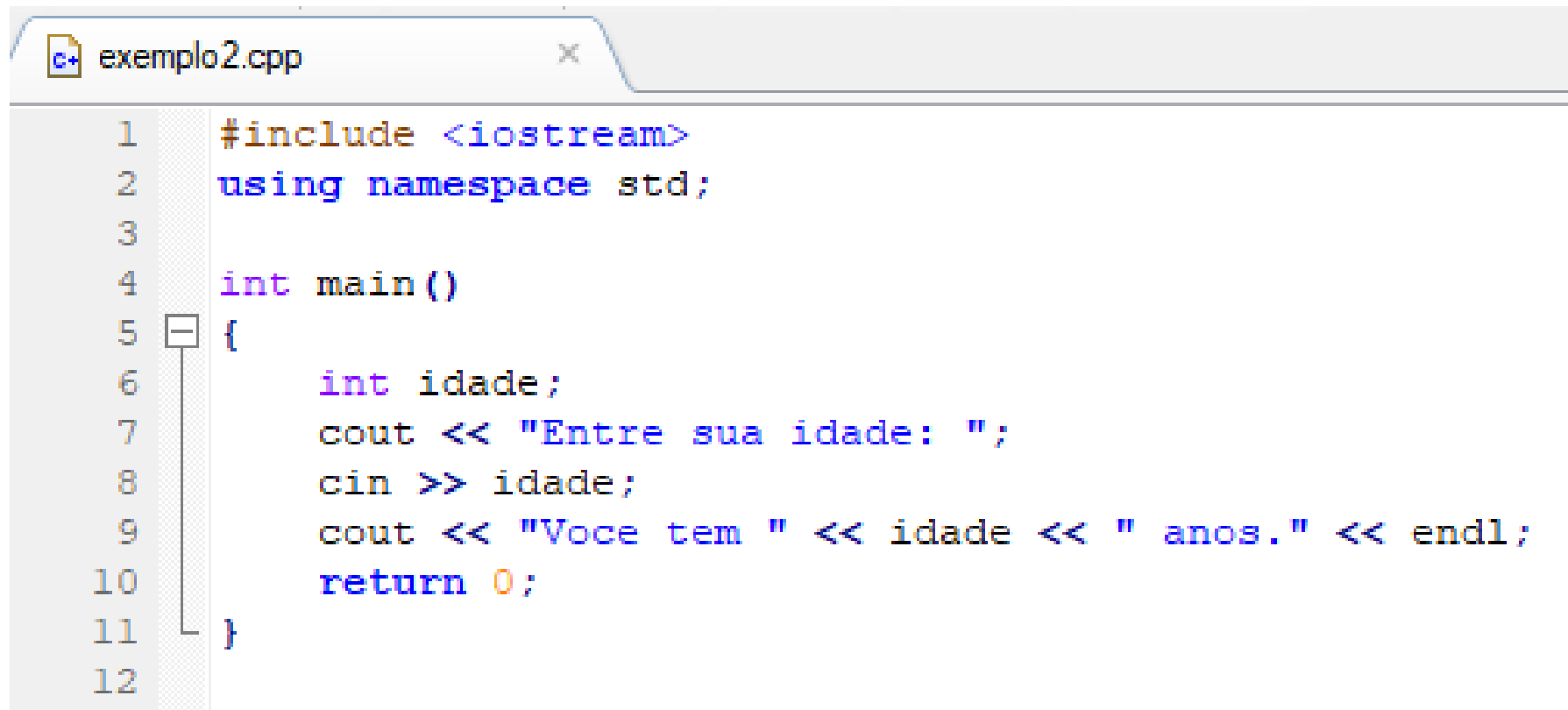
# Exibindo informações na tela: **cout**

## exemplo1.cpp

```
exemplo1.cpp
1  #include <iostream> /* inclui diretivas de entrada-saída */
2
3
4  //utiliza o namespace std para definir todas as funções da biblioteca padrão
5  using namespace std;
6
7  //declaração de constante
8  #define PRECO 1.99
9
10 int main()
11 {
12     //declarando as variáveis
13     int pera = 3;
14     char qualidade = 'A';
15     float peso = 2.5;
16
17     //saída de dados (cout)
18     cout << "Existem " << pera << " peras de qualidade " << qualidade
19         << " pesando " << peso << " quilos." << endl;
20     cout << "O preco por quilo eh R$" << PRECO
21         << ", o total em R$ eh " << peso * PRECO << endl;
22
23     return 0;
24 }
```

# Lendo informação: **cin**

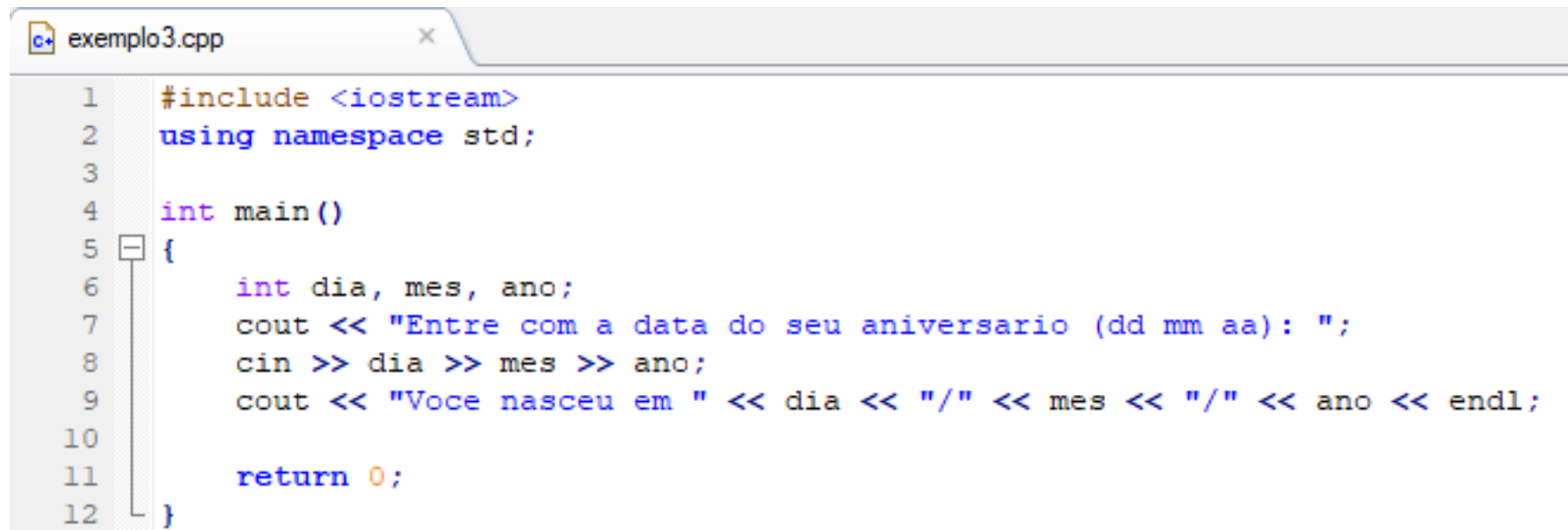
## exemplo2.cpp



```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int idade;
7      cout << "Entre sua idade: ";
8      cin >> idade;
9      cout << "Voce tem " << idade << " anos." << endl;
10     return 0;
11 }
12
```

# Lendo informação: **cin**

## exemplo3.cpp



```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int dia, mes, ano;
7      cout << "Entre com a data do seu aniversario (dd mm aa): ";
8      cin >> dia >> mes >> ano;
9      cout << "Voce nasceu em " << dia << "/" << mes << "/" << ano << endl;
10
11     return 0;
12 }
```

# Operações Aritméticas

- Em C++, pode-se executar operações aritméticas usando variáveis e constantes. Algumas operações mais comuns são:

+ adição

- subtração

\* multiplicação

/ divisão

% resto (módulo)

# exemplo4.cpp

```
exemplo4.cpp x
1  /* programa que calcula o perímetro e a área de uma
2  circunferência de raio R (fornecido pelo usuário) */
3
4  #include <iostream> // inclui diretivas de entrada-saída
5  #include <cmath> // inclui diretivas das funções matemáticas */
6
7  using namespace std;
8  #define PI 3.14159
9
10 int main( )
11 {
12
13     int raio;
14     float perim, area;
15
16     cout << "Entre com o valor do raio: ";
17     cin >> raio;
18
19     perim = 2 * PI * raio;
20     area = PI * pow(raio, 2);
21
22     cout << "O perimetro da circunferencia de raio " << raio
23         << " eh " << perim << endl;
24     cout << "e a area eh " << area << endl;
25
26     return 0;
27 }
```



# Operadores Relacionais

- Em C++ , há operadores que podem ser usados para comparar expressões: os operadores relacionais.

< menor que

> maior que

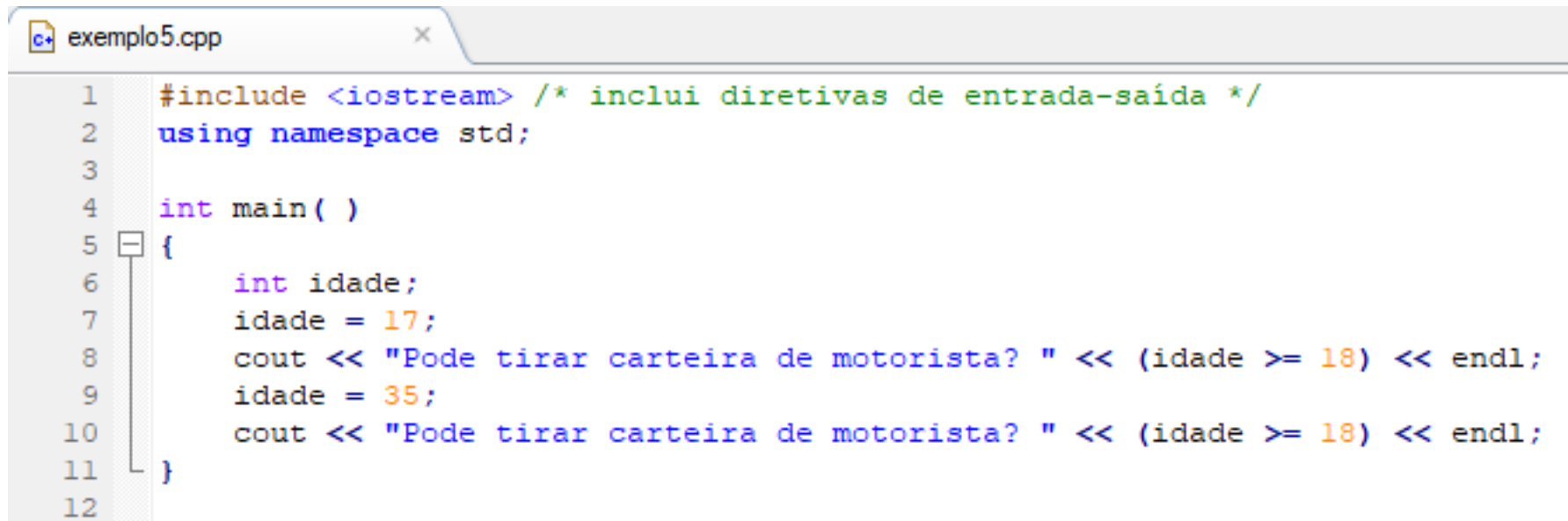
<= menor ou igual que ( $\leq$ )

>= maior ou igual que ( $\geq$ )

== igual a

!= não igual a ( $\neq$ )

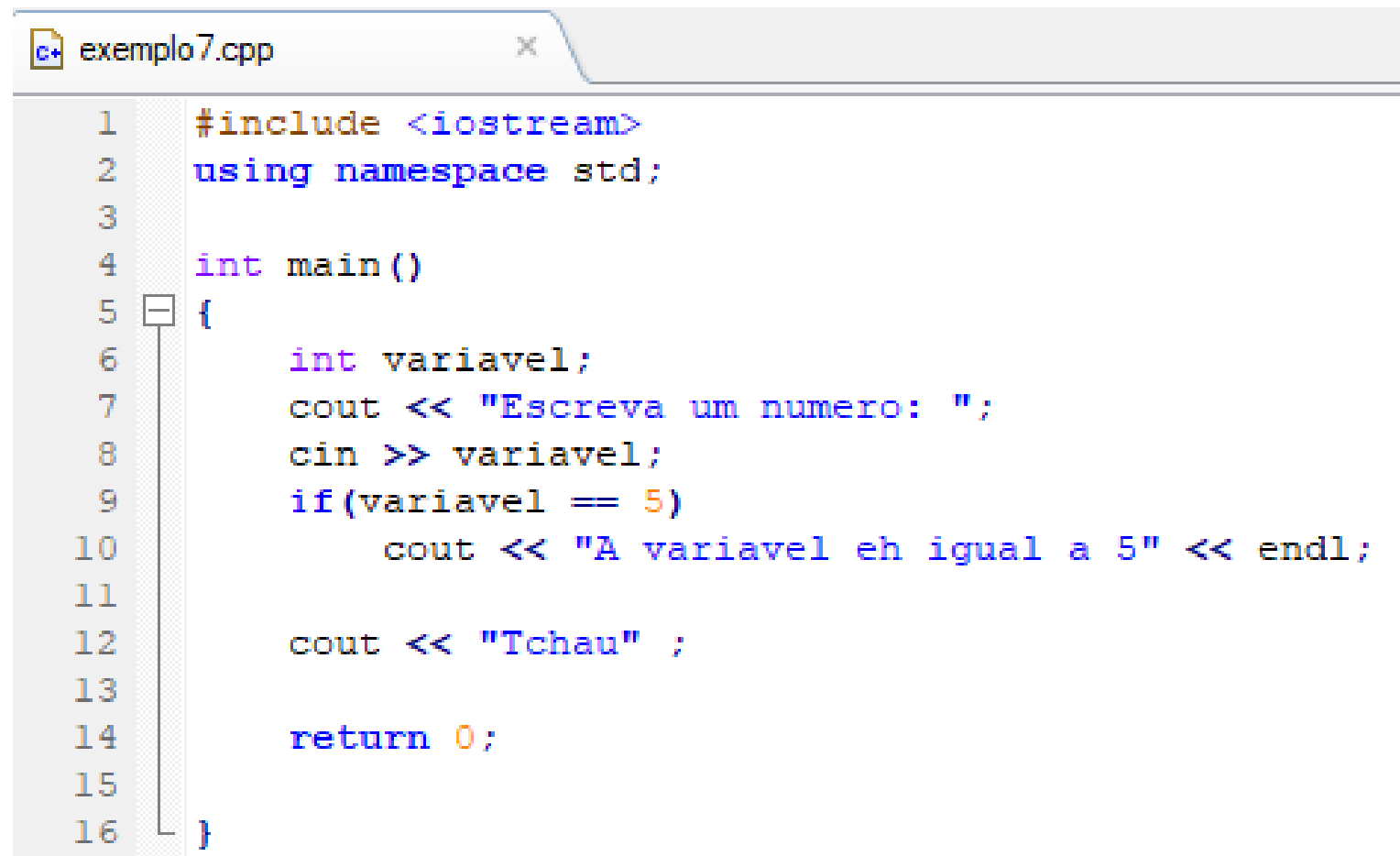
# exemplo5.cpp



```
1  #include <iostream> /* inclui diretivas de entrada-saída */
2  using namespace std;
3
4  int main( )
5  {
6      int idade;
7      idade = 17;
8      cout << "Pode tirar carteira de motorista? " << (idade >= 18) << endl;
9      idade = 35;
10     cout << "Pode tirar carteira de motorista? " << (idade >= 18) << endl;
11 }
12
```

# Estruturas de Decisão em C++

- **Simples:** exemplo6.cpp

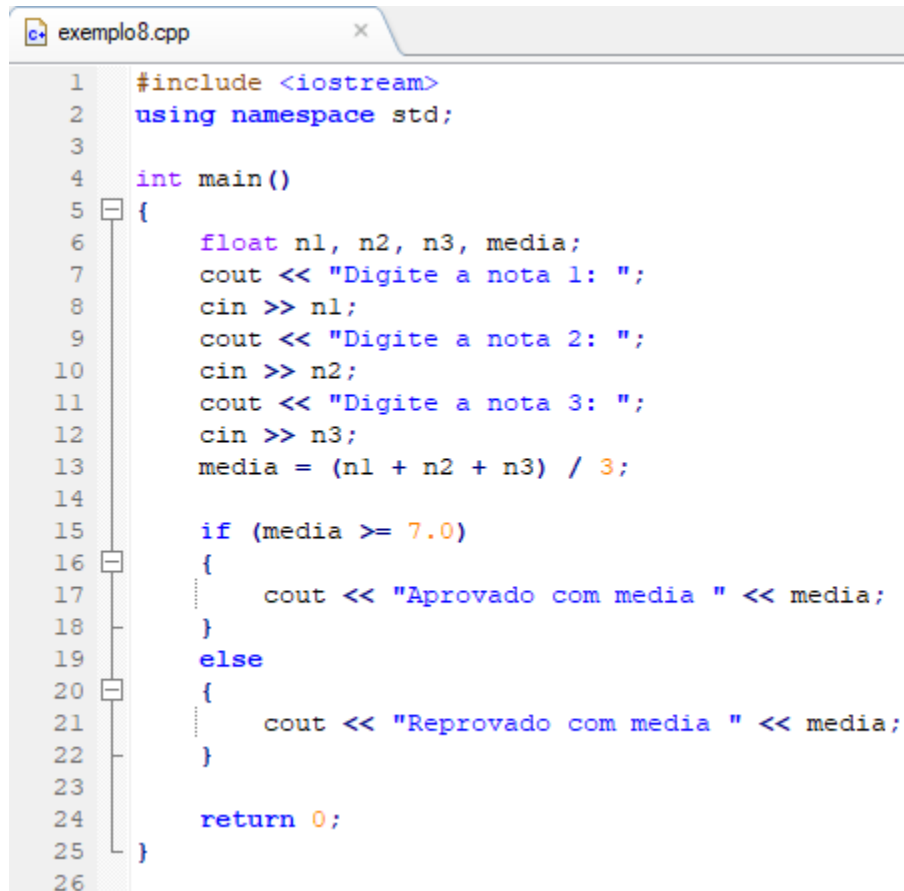


The image shows a code editor window with a single tab titled 'exemplo7.cpp'. The code is written in C++ and demonstrates a simple decision structure using an if statement. The code is as follows:

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int variavel;
7      cout << "Escreva um numero: ";
8      cin >> variavel;
9      if(variavel == 5)
10         cout << "A variavel eh igual a 5" << endl;
11
12     cout << "Tchau" ;
13
14     return 0;
15
16 }
```

# Estruturas de Decisão em C++

- **Composta:** exemplo7.cpp



```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      float n1, n2, n3, media;
7      cout << "Digite a nota 1: ";
8      cin >> n1;
9      cout << "Digite a nota 2: ";
10     cin >> n2;
11     cout << "Digite a nota 3: ";
12     cin >> n3;
13     media = (n1 + n2 + n3) / 3;
14
15     if (media >= 7.0)
16     {
17         cout << "Aprovado com media " << media;
18     }
19     else
20     {
21         cout << "Reprovado com media " << media;
22     }
23
24     return 0;
25 }
```

# Estruturas de Decisão em C++

- Encadeada: exemplo8.cpp

```
exemplo9.cpp x
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      float n1, n2, n3, media, freq;
7      cout << "Digite a frequencia: ";
8      cin >> freq;
9      cout << "Digite a nota 1: ";
10     cin >> n1;
11     cout << "Digite a nota 2: ";
12     cin >> n2;
13     cout << "Digite a nota 3: ";
14     cin >> n3;
15     media = (n1 + n2 + n3) / 3;
16     if (freq >= 75)
17     {
```

# Estruturas de Decisão em C++

- Encadeada: exemplo9.cpp

```
18     if (media >= 7.0)
19     {
20         cout << "Aprovado com media " << media << " e frequencia " << freq;
21     }
22     else if(media >= 3)
23     {
24         cout << "Exame";
25     }
26     else
27     {
28         cout << "Reprovado";
29     }
30 }
31 else
32 {
33     cout << "Reprovado por falta";
34 }
35 return 0;
36 }
```

# exemplo10.cpp

```
exemplo10.cpp x
1  #include <iostream>
2  #include <iomanip>
3  // necessario para usar setw() e setf() em cout
4
5  using namespace std;
6  int main()
7  {
8      float n1, n2, n3, soma;
9      cout << "Digite o numero 1 ";
10     cin >> n1;
11     cout << "Digite o numero 2 ";
12     cin >> n2;
13     cout << "Digite o numero 3 ";
14     cin >> n3;
15
16     soma = n1 + n2 + n3;
17
18     cout << "Soma = " << soma << endl;
19     cout.setf (ios::fixed | ios::showpoint); // reais em ponto fixo
20     cout.precision(2); // 2 casa decimais
21     //setw(8) fixa tamanho da representação em 8 digitos
22     cout << "Media = " << setw(8) << soma / 3.0 << endl;
23     cout << "Produto = " << n1 * n2 * n3 << endl;
24
25     return 0;
26 }
```

# Operadores Lógicos

- Os operadores lógicos são:
  - ! NÃO lógico, operação de negação (operador unário)
  - && E lógico, conjunção (operador binário)
  - || OU lógico, disjunção (operador binário).



# exemplo11.cpp

exemplo11.cpp

```
1  /*
2
3  O programa verifica se as três variáveis lado1, lado2, e lado3, podem ser lados
4  de um triângulo reto. Nós usamos o fato que os três valores devem ser positivos,
5  e que o quadrado de um dos lados deve ser igual a soma dos quadrados dos outros
6  lados (Teorema de Pitágoras) para determinar se o triângulo é reto.
7
8  */
9
10 #include <iostream>
11 using namespace std;
12 int main()
13 {
14     int lado1, lado2, lado3;
15     int s1, s2, s3;
16
17     cout << "Entre com o tamanho do lado 1: ";
18     cin >> lado2;
19     cout << "Entre com o tamanho do lado 2: ";
20     cin >> lado3;
21     cout << "Entre com o tamanho do lado 3: ";
22     cin >> lado1;
```

# exemplo11.cpp

```
23
24 // calcula o quadrado dos lados
25 s1 = lado1 * lado1;
26 s2 = lado2 * lado2;
27 s3 = lado3 * lado3;
28
29 // testa a condicao para um triangulo reto
30 if (lado1 > 0 && lado2 > 0 && lado3 > 0 )
31 {
32     if (s1 == s2 + s3 || s2 == s1 + s3 || s3 == s1 + s2)
33     {
34         cout << "Triangulo reto!" << endl;
35     }
36     else
37     {
38         cout << "Nao pode ser um triangulo!" << endl;
39     }
40 }
41 else
42 {
43     cout << "Nao pode ser um triangulo!" << endl;
44 }
45
46 return 0;
47 }
```

# O comando Switch - exemplo12.cpp

```
*exemplo12.cpp
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int num1, num2, resultado;
7      float div;
8      char op; // operação
9
10     cout << "Digite o primeiro numero: ";
11     cin >> num1;
12     cout << "Digite o segundo numero: ";
13     cin >> num2;
14     cout << "Digite a operacao: " << endl;
15     cout << "[+]" << endl;
16     cout << "[-]" << endl;
17     cout << "[*]" << endl;
18     cout << "[/]" << endl;
19     cin >> op;
20
```

# O comando Switch - exemplo12.cpp

```
21     switch(op)
22     {
23     case '+':
24         resultado = num1 + num2;
25         cout << "Soma: " << resultado << endl;
26         break;
27     case '-':
28         resultado = num1 - num2;
29         cout << "Subtracao: " << resultado << endl;
30         break;
31     case '*':
32         resultado = num1 * num2;
33         cout << "Multiplicacao: " << resultado << endl;
34         break;
35     case '/':
36         if(num2 != 0)
37         {
38             div = (float)num1 / num2;
39             cout << "Divisao: " << div << endl;
40         }
41         else
42         {
43             cout << "Divisao por zero!" << endl;
44         }
45         break;
46     default:
47         cout << "Operacao inexistente." << endl;
48         break;
49     }
50
51     return 0;
52 }
```

# Exercícios Propostos

1. Elabore um programa em C++ que solicite o peso e a altura de uma determinada pessoa. Após a digitação, exibir se esta pessoa está ou não com seu peso ideal, conforme tabela abaixo:

IMC ( $IMC = peso / altura^2$ )	MENSAGEM
$imc < 20$	Abaixo do peso
$20 \geq imc < 25$	Peso Ideal
$IMC \geq 25$	Acima do Peso

2. Elaborar um programa em C++ em que dada a idade de um nadador, classifique-o em uma das seguintes categorias: infantil A (de 5 a 7 anos), infantil B (de 8 a 10 anos), juvenil A (de 11 a 13 anos), juvenil B (14 a 17 anos) e senior (maior que 17 anos)

# Exercícios Propostos

3. Faça um programa em C++ que receba o número de horas trabalhadas e o valor do salário mínimo. Calcule e mostre o salário a receber seguindo as regras abaixo:
  - a. A hora trabalhada vale a metade do salário mínimo;
  - b. O salário bruto equivale ao número de horas trabalhadas multiplicado pelo valor da hora trabalhada
  - c. O imposto equivale a 3% do salário bruto;
  - d. O salário a receber equivale ao salário bruto menos o imposto.
  
4. Construa um programa em C++ que calcule o novo salário de um funcionário. Considere que o funcionário deverá receber um reajuste de 15% caso seu salário seja menor que 1000. Se o salário for maior ou igual a 1000, mas menor ou igual a 1500, o reajuste deve ser de 10%. Caso o salário seja maior que 1500, o reajuste deve ser de 5%.

# Exercícios Propostos

3. Faça um programa em C++ que receba o número de horas trabalhadas e o valor do salário mínimo. Calcule e mostre o salário a receber seguindo as regras abaixo:
  - a. A hora trabalhada vale a metade do salário mínimo;
  - b. O salário bruto equivale ao número de horas trabalhadas multiplicado pelo valor da hora trabalhada
  - c. O imposto equivale a 3% do salário bruto;
  - d. O salário a receber equivale ao salário bruto menos o imposto.
  
4. Construa um programa em C++ que calcule o novo salário de um funcionário. Considere que o funcionário deverá receber um reajuste de 15% caso seu salário seja menor que 1000. Se o salário for maior ou igual a 1000, mas menor ou igual a 1500, o reajuste deve ser de 10%. Caso o salário seja maior que 1500, o reajuste deve ser de 5%.

# Exercícios Propostos

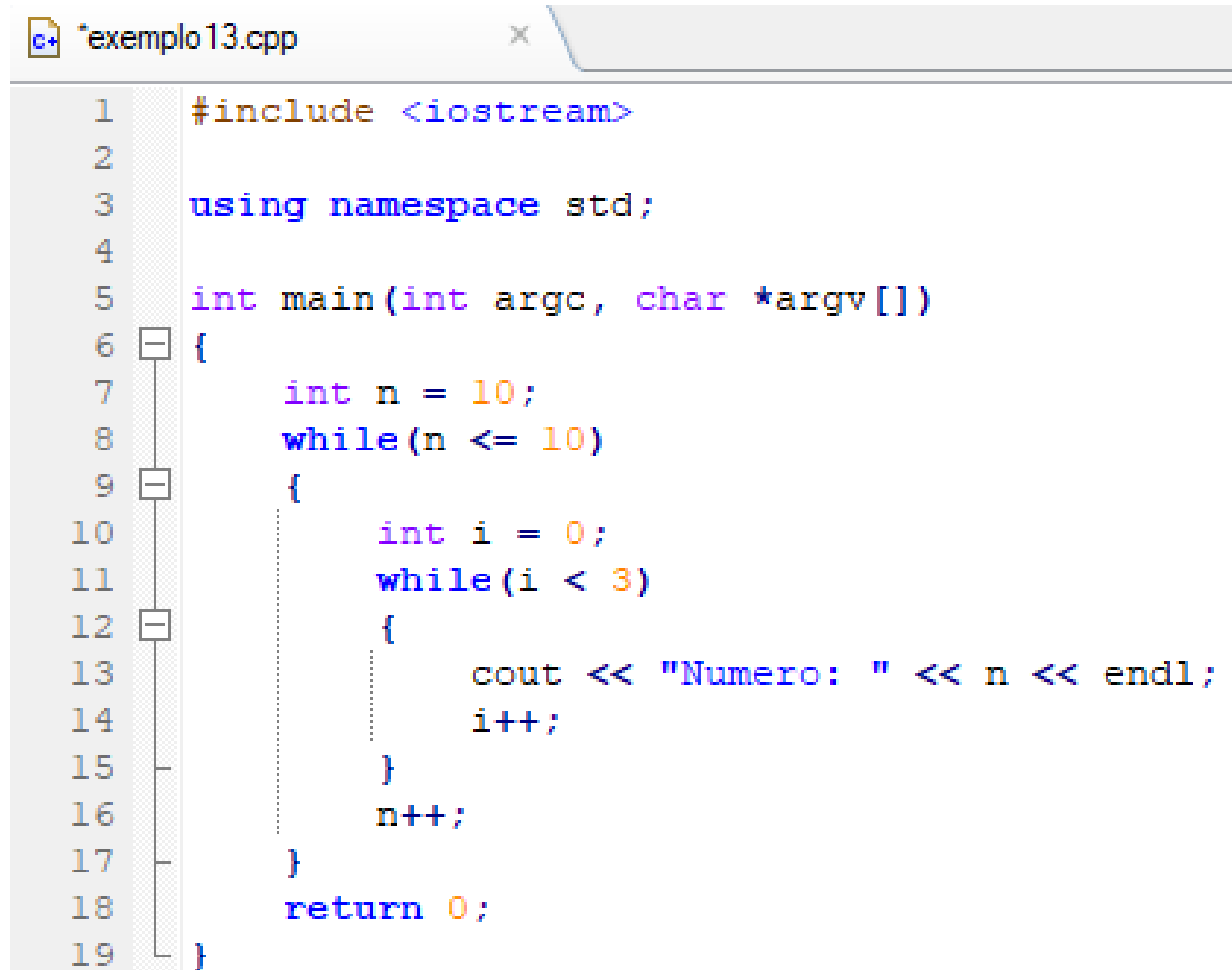
5. Construa um programa em C++ que calcule e apresente quanto deve ser pago por um produto considerando a leitura do preço de etiqueta (PE) e o código da condição de pagamento (CP). Utilize para os cálculos a tabela de condições de pagamento a seguir:

Código da condição de pagamento	Condição de pagamento
1	À vista em dinheiro ou cheque, com 10% de desconto
2	À vista com cartão de crédito, com 5% de desconto
3	Em 2 vezes, preço normal de etiqueta sem juros
4	Em 3 vezes, preço de etiqueta com acréscimo de 10%



# Estruturas de Repetição em C++

## while – exemplo13.cpp

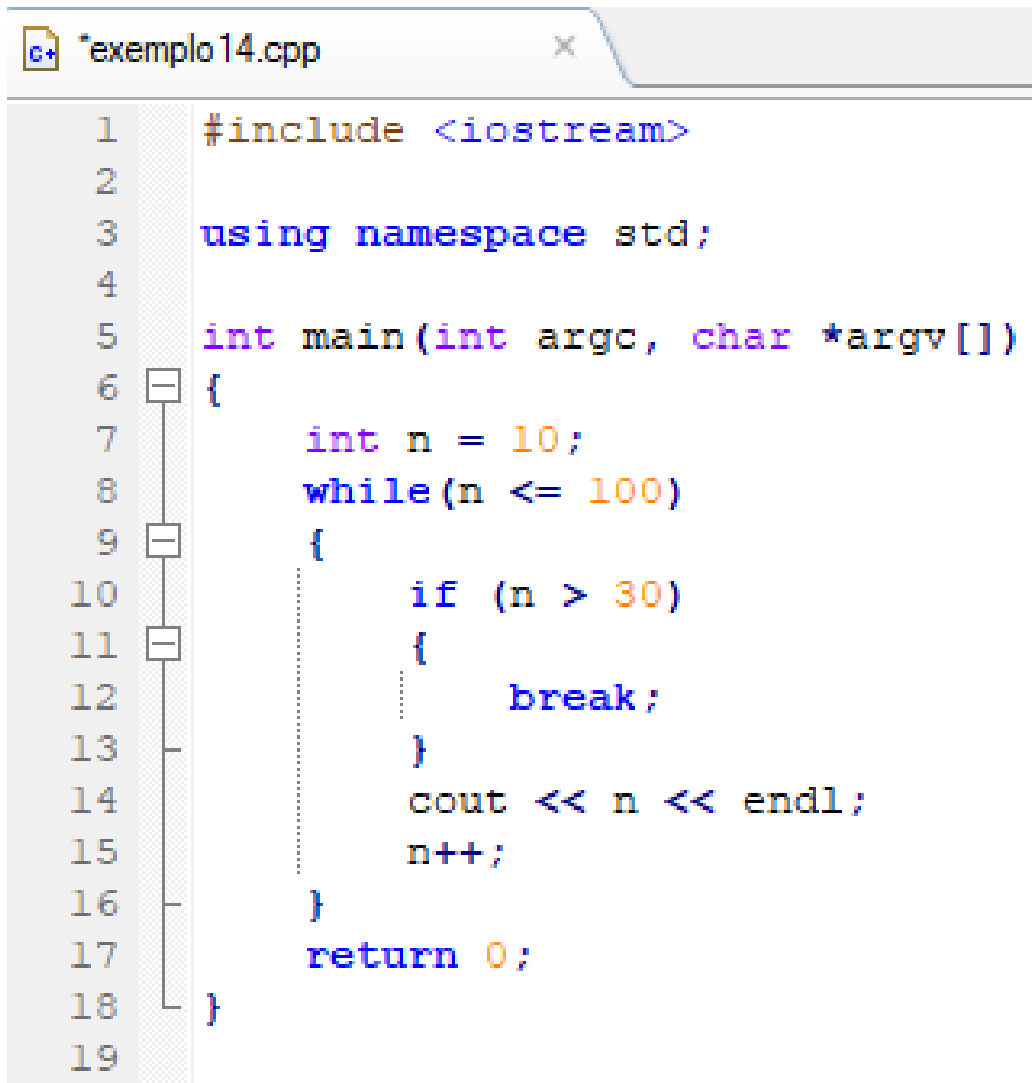


The image shows a code editor window with a single file named "exemplo13.cpp". The code is written in C++ and demonstrates a nested while loop. The first while loop iterates as long as 'n' is less than or equal to 10. Inside this loop, there is a second while loop that iterates as long as 'i' is less than 3. In each iteration of the inner loop, the program prints "Numero: " followed by the value of 'n' and a newline character. After the inner loop finishes, 'n' is incremented by 1. The outer loop also increments 'n' by 1 after each iteration. The program returns 0 at the end.

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main(int argc, char *argv[])
6  {
7      int n = 10;
8      while(n <= 10)
9      {
10         int i = 0;
11         while(i < 3)
12         {
13             cout << "Numero: " << n << endl;
14             i++;
15         }
16         n++;
17     }
18     return 0;
19 }
```

# Estruturas de Repetição em C++

## while – exemplo14.cpp



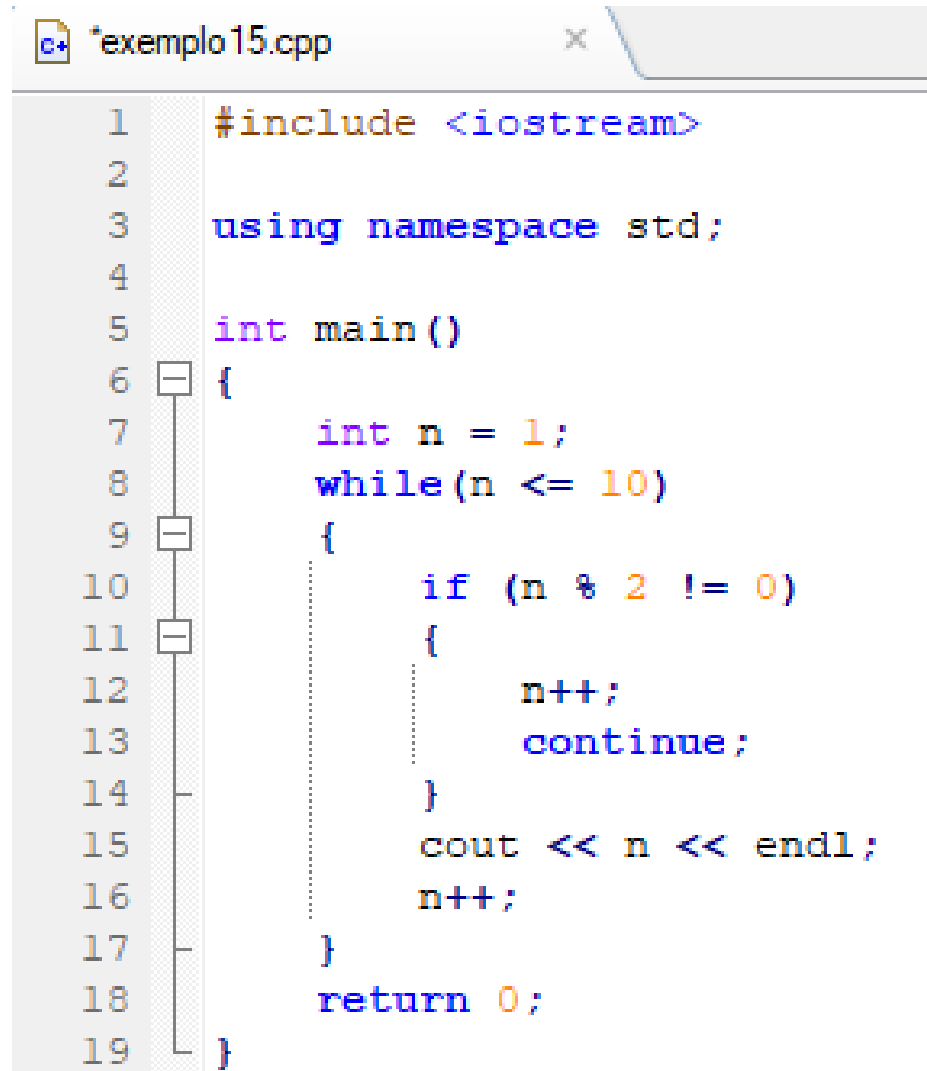
The image shows a code editor window titled "exemplo14.cpp". The code is written in C++ and demonstrates a while loop. The code is as follows:

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main(int argc, char *argv[])
6  {
7      int n = 10;
8      while(n <= 100)
9      {
10         if (n > 30)
11         {
12             break;
13         }
14         cout << n << endl;
15         n++;
16     }
17     return 0;
18 }
19
```

The code is color-coded: keywords like `include`, `using`, `int`, `while`, `if`, `break`, `return`, and `cout` are in blue; identifiers like `n` and `argc` are in purple; and literals like `10`, `100`, and `30` are in orange. The code is displayed in a monospaced font. On the left side of the code, there is a vertical line with square markers at lines 6, 9, 11, and 16, indicating the structure of the code blocks (main function, while loop, if statement, and return statement).

# Estruturas de Repetição em C++

## while – exemplo15.cpp



The image shows a code editor window titled "exemplo15.cpp". The code is written in C++ and uses a while loop to print odd numbers from 1 to 10. The code is as follows:

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      int n = 1;
8      while(n <= 10)
9      {
10         if (n % 2 != 0)
11         {
12             n++;
13             continue;
14         }
15         cout << n << endl;
16         n++;
17     }
18     return 0;
19 }
```

The code is color-coded: keywords like `include`, `using namespace`, `while`, `if`, `continue`, `cout`, and `return` are in blue; identifiers like `n` are in purple; and literals like `1`, `10`, `0`, and `endl` are in orange. The editor has a line number margin on the left and a vertical scrollbar on the right.

# Estruturas de Repetição em C++

## for – exemplo16.cpp

```
exemplo16.cpp x
1  /*
2      Cálculo do fatorial
3      0! = 1! = 1
4      3! = 3 * 2 * 1 = 6
5      4! = 4 * 3 * 2 * 1 = 24
6      5! = 5 * 4 * 3 * 2 * 1 = 120
7  */
8
9  #include <iostream>
10
11 using namespace std;
12
13 int main(int argc, char *argv[])
14 {
15     int num = 4, fat = 1;
16
17     for(int i = 1; i < num ; i++)
18         fat = fat * (i + 1);
19     cout << "Fatorial: " << fat << endl;
20     return 0;
21 }
```

# Estruturas de Repetição em C++

## do..while – exemplo17.cpp

```
exemplo17.cpp x
1  #include <iostream>
2  #include <cstdlib>
3
4  using namespace std;
5  int main()
6  {
7      int nr = 0;
8      do
9      {
10         cout << "Digite o numero: ";
11         cin >> nr;
12
13         if (nr % 2 == 0)
14         {
15             cout << "Este numero eh par" << endl;
16         }
17         else
18         {
19             cout << "Este numero eh impar" << endl;
20         }
21
22         system("pause");
23         system("cls");
24     }
25     while (nr != 0);
26     return 0;
27 }
28 }
```

# Exercícios Propostos

6. Escreva um programa em C++ que mostre o quadrado dos números inteiros no intervalo de 1 a 20.
7. Escreva um programa em C++ que escreva todos os números múltiplos de 5, no intervalo de 1 a 500.
8. Em uma eleição presidencial existem dois candidatos. Os votos são informados através de códigos. Os dados utilizados para a contagem dos votos têm-se a seguinte codificação: 1,2= voto para os respectivos candidatos; 3= voto nulo; 4= voto em branco; Elabore um programa em C++ que leia o código do candidato em um voto. Como finalizador do conjunto de votos, tem-se o valor 0. Calcule e escreva: (1) percentual de votos para cada candidato; (2) percentual de votos nulos; (3) percentual de votos em branco;
9. Faça um programa em C++ que leia dez conjuntos de dois valores, o primeiro representando o número do aluno e o segundo a sua altura em centímetros. Encontre o aluno mais alto e o mais baixo. Exiba o número do aluno mais baixo o número de aluno mais alto e as respectivas alturas.

# Exercícios Propostos

10. Em um cinema, certo dia, cada espectador respondeu a um questionário, que perguntava a sua idade (ID) e a opinião em relação ao filme (OP), seguindo os seguintes critérios:

Opinião (OP)	Significado
1	Ótimo
2	Bom
3	Regular
4	Ruim

Ao final da pesquisa será indicado quando a idade do usuário for informada como negativa (idade inexistente). Construa um programa em C++ que, lendo esses dados, calcule e apresente:

- A. Quantidade de pessoas que respondeu a pesquisa
- B. Média de idade das pessoas que responderam a pesquisa
- C. Porcentagem de cada uma das respostas

# Referência desta aula

- Notas de Aula do Prof. Prof. Armando Luiz N. Delgado baseado em revisão sobre material de Prof.a Carmem Hara e Prof. Wagner Zola.
- <http://www.cplusplus.com/reference/>

Obrigado

Rodrigo