Lista

Universidade Federal de Santa Catarina - Prof. Gian

Atenção: Nos exercícios abaixo os códigos apresentados nem sempre estão completos. Cabe ao aluno finalizar o código sugerido. Nos execícios que contêm ponteiros, refaçam-os usando ponteiros inteligentes - C++14.

1. Qual é a diferença entre declaração de classe e uma declaração de um objeto (também conhecido como instanciação)?

2. O que é uma variável de instância?

3. Quais são os aspectos básicos de uma classe?

4. O que significa os modificadores `const` e `static` em C++? Como posso combiná-los para criar um símbolo constante?

5. O que significa o termo `public`? E o termo `private`?

6. O que é um construtor?

7. O que é um método "accessor"? Qual é a vantagem em utilizar esse método ao invés de prover acesso direto ao atributo?

8. O que é um "mutator", ou *setter* método?

9. Imprima na tela o seu nome 100 vezes.

```
class ImprimeNome {
  public:
  void imprime( ) {
    for( int contador = 0; contador < 100; contador++) {
      cout << " Meu Nome " ;
    }
  }
}

int main(){
  ImprimeNome* imp = new ImprimeNome();
  imp->imprime();
  delete imp;
}
```

Compile e execute essa Classe.

```
MeuNome/exercicios-poo$g++ ImprimeNome.cpp -o imprime
MeuNome/exercicios-poo$./imprime
```

10. Imprima na tela os números de 1 até 100.

```
class ImprimeNome {
  public:
  void imprime( ) {
    for( int contador = 0; contador < 100; contador++) {
      cout << contador << endl ;
    }
  }
}

int main(){
  ImprimeNome* imp = new ImprimeNome();
  imp->imprime();
  delete imp;
}
```

Compile e execute essa Classe.

```
MeuNome/exercicios-poo$g++ ImprimeAte100.cpp -o imprime100
MeuNome/exercicios-poo$./imprime100
```

11. Faça um programa que percorra todos os número de 1 até 100. Para os números ímpares, deve ser impresso um "*", e para os números pares, deve ser impresso dois "**". Veja o exemplo abaixo:

```
*
**
*
**
```

```
class ImprimePadrao1 {
  public:
  void imprime( ) {
    for(int contador = 1; contador <= 100; contador++) {
      int resto = contador % 2;
      if ( resto == 1) {
        cout << "* " <<endl;
      } else {
        cout << "**" <<endl;
      }
    }
  }
}

int main(){
  ImprimePadrao1* imp = new ImprimePadrao1();
  imp->imprime();
  delete imp;
}
```

Compile e execute essa Classe.

```
MeuNome/exercicios-poo$g++ ImprimePadrao1.cpp -o imprimep1
MeuNome/exercicios-poo$./imprimep1
```

12. Faça um programa que percorra todos os número de 1 até 100. Para os números múltiplos de 4, imprima a palavra "PI", e para os outros, imprima o próprio número. Veja o exemplo abaixo:

```
1
2
3
PI
5
6
7
PI
```

```
class ImprimePadrao2 {
  public:
  static void imprime( ) {
    for( int contador = 1; contador <= 100; contador ++) {
      int resto = contador % 4;
      if( resto == 0) {
        cout << "PI"<<endl;
      } else {
        cout << contador<< endl;
      }
    }
  }
}

int main(){
  ImprimePadrao2* imp = new ImprimePadrao2();
  imp->imprime();
  delete imp;
}
```

Compile e execute essa Classe.

```
MeuNome/exercicios-poo$g++ ImprimePadrao2.cpp -o imprimep2
MeuNome/exercicios-poo$./imprimep2
```

13. Crie um programa que imprima na tela um triângulo de "*". Veja o exemplo abaixo:

```
*
**
***
****
*****
```

14. Crie um programa que imprima na tela vários triângulos de "*". Observe o padrão abaixo.

```
*
**
***
****
*
**
***
****
```

15. Os números de Fibonacci são uma sequência de números definida recursivamente. O primeiro elemento da sequência é 0 e o segundo é 1. Os outros elementos são calculados somando os dois antecessores.

```
0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233...
```

Crie uma classe para imprimir os N primeiros números da sequência de Fibonacci. N é um parâmetro do construtor.

16. Use seus conhecimentos para criar um programa que mostre um menu de atalho para os 5 padrões que acabamos de fazer. Exemplo:

```
$ ./GeradorDePadroes
Gerador de Padrões
Escolha a opção desejada :
1- Padrão 1
2- Padrão 2
3- Padrão 3
4- Padrão 4
5- Padrão 5
0- Sair
```

Se digitar o numero 1, ele automaticamente tem de executar o código para o padrão 1, e assim sucessivamente.

17. Quais seriam os atributos e métodos de um caixa de banco automático.

18. Como descrever uma lâmpada que está à venda em um supermercado. Que dados devem ser atributos para essa classe?

19. Imagine uma lâmpada que possa ter três estados: apagada, acesa e meia-luz. Usando a classe "Lâmpada" como base, escreva a classe "LampadaTresEstados".

20. Inclua, na classe "Lâmpada", uma operação "estaLigada" que retorne verdadeiro se a lâmpada estiver ligada e falso, caso contrário.

21. Crie uma classe Livro que represente os dados básicos de um livro, sem se preocupar com a sua finalidade.

22. Usando o resultado do exercício anterior como base, crie uma classe "LivroDeLivraria" que represente os dados básicos de um livro que está à venda em uma livraria.

23. Usando o resultado da classe "Livro" como base, crie uma classe "LivroDeBiblioteca" que represente os dados básicos de um livro de uma biblioteca, que pode ser emprestado a leitores.

24. Crie uma classe (atributos e métodos) para representar uma entrada de cinema.

25. Crie uma classe Professor com os atributos: nome do professor, nome do departamento, data de admissão, número de registro. Inclua na classe um construtor para definir os dados e uma operação para imprimir o conteúdo.

26. Crie uma classe Data com os atributos: dia, mês e ano.

27. Reescreva a classe Professor de forma que a data de admissão seja um atributo do tipo Data.

28. Modifique a classe Data criado anteriormente para conter uma operação de construção capaz de definir o dia, mês e ano e uma operação capaz de imprimir a data.

29. Modifique a classe Data para informar se:

1. a data é de um ano bissexto;
2. a data é valida (considerando ano bissexto, em que mês está e em qual dia).

30. Modifique a classe Data para sobrecarregar os seguintes operadores:

++ incrementa um dia;

- - decrementa um dia;

\- diferença em dias; e

== se as datas são iguais.

31. Um contador digital é um contador-limitado que reinicia quando seu valor inteiro atinge um certo valor máximo. Exemplos incluem os números no relógio digital e o odômetro em um carro. Defina uma classe para um contador-limitado. Essa classe deve ter a habilidade de definir o valor mínimo e máximo, incrementar o contador e retornar o valor atual do contador.

32. Defina uma classe para uma fração, um número racional composto de dois valores inteiros. Defina, também, métodos para adição, subtração, multiplicação e divisão de frações. Como você irá gerenciar as reduções das frações para um mínimo divisor comum? Implemente os métodos relacionados com operadores matemáticos com a sobrecarga de operadores ($+$,$-$,$*$ e $/$).

33. Defina uma classe para os números complexos. Defina métodos para adição, subtração e multiplicação de números complexos. Implemente os métodos relacionados com operadores matemáticos com a sobrecarga de operadores ($+$,$-$ e $*$).

34. Considere as duas combinações abaixo de classes e funções em C++. Explique a diferença para um usuário(desenvolvedor) em usar a função `addi`.

```
class example1 {
public:
  int i;
};

int addi(example1 & x, int j){
  x.i = x.i + j;
  return x.i;
}

class example2 {
private:
  int i;
public:
  int addi(int j)
    { i = i + j; return i;}
};
```

35. Apresentada a seguinte definição de classe:

```
class CreateDestroy
{
public:
   CreateDestroy() { cout << "constructor called, "; }
   ~CreateDestroy() { cout << "destructor called, "; }
};
```

Qual será a saída do programa?

```
int main()
{
   CreateDestroy c1;
   CreateDestroy c2;
   return 0;
}
```

36. Crie uma classe `Rectangle`. Ela deve possuir os atributos `altura` e `largura`, cada um deles possui o valor padrão definido como 1. Essa classe deve ter métodos que calculam o perímetro e a área do retângulo. Deve possuir os métodos get e set para os atributos `altura` e `largura`. Os métodos set devem verificar se os valores passados como parâmetro estão dentro dos limites [0;20]. Escreva um programa para testar/usar a classe `Rectangle`.

37. Crie uma classe `Rectangle` mais sofisticada que o exercício anterior. Ela deve armazenar as coordenadas cartesianas (bidimensionais) dos quatro vértices do retângulo. Os métodos set devem verificar se as coordenadas informadas especificam de fato um retângulo válido (e se os valores das coordenadas estão entre 0 e 20.0). O comprimento é a maior das duas dimensões. Inclua um método isSquare que verifica se o retângulo é um quadrado.

38. Rescreva o exercício anterior, e no lugar de coordenadas (x,y), utilize uma classe que represente esses pontos cartesianos. Nome da classe será `Point`.

39. Em noites sem nuvens pode-se, muitas vezes, observar pontos brilhantes no céu que se deslocam com grande velocidade, e em poucos segundos desaparecem de vista: são as chamadas estrelas cadentes, ou meteoros. Meteoros são na verdade partículas de poeira de pequenas dimensões que, ao penetrar na atmosfera terrestre, queimam-se rapidamente (normalmente a uma altura entre 60 120 quilômetros). Se os meteoros são suficientemente grandes, podem não queimar-se completamente na atmosfera e dessa forma atingem a superfície terrestre: nesse caso são chamados de meteoritos.
Zé Felício tem muitas fazendas e adora astronomia. Ele não sabe qual a quantidade de meteoritos caem por ano e contrata você para fazer um programa para simular a possibilidade de quedas nas suas propriedades.
Crie uma classe `Fazenda` que armazene um retângulo para delimitar a área. As linhas que delimitam a fazenda são paralelas aos eixos cartesianos. Essa classe deve ter um método que recebe um ponto no espaço cartesiano e verifique se o objeto (ponto) está dentro da fazenda.
Crie um programa que peça para o usuário informar a taxa de queda de meteoritos (quantidade anual) e exiba uma lista com o nome da fazenda e a quantidade de meteoritos (simulados) que caíram nessa propriedade. Gere aleatoriamente as posições e verifique quantos caíram dentro de alguma das fazendas. As informações sobre as fazendas (nome e área) são definidas em uma classe que você irá criar para usar a classe `Fazenda`.