Iniciada	sexta, 12 de fevereiro de 2021 às 09:57
Estado	Terminada
Terminada em	sexta, 12 de fevereiro de 2021 às 10:37
Tempo gasto	40 minutos 1 segundo

Pergunta 1

Respondida

Nota: 20,00

Considere a classe seguinte:

```
class Bingo {
   const string data, nome;
   int quantos_maximo, limite_inf, limite_sup;
   ... sorteados;  //representa o conjunto de nº sorteados
   ...             //sem repetições
};
```

A classe Bingo regista os números inteiros que saíram (não os sorteia propriamente) sorteados numa dada data e para um dado evento representado por um nome. Os números aceites deverão: ser únicos, ou seja, sem estar repetidos; pertencer ao intervalo {limite_inf, limite_sup} e não ultrapassar a quantidade máxima (quantos_maximo). Pode acrescentar coisas à classe, mas não retirar nem mudar o que já existe.

Faça com que seja possível:

- Criar objetos indicando todos os dados. O conjunto de números sorteados é o único opcional - se não indicado é inicialmente vazio:
  const Bingo bingo1("Académica de Coimbra","31 de janeiro de 2020",10, 1, 99, {1, 16, 48, 49, 57, 59, 65, 84, 90, 99});
  Bingo bingo2("Belenenses", "12 de fevereiro de 2021", 40, 1, 50); //conjunto inicialmente vazio, aceita valores
  //entre 1 e 50, e pode ter até 40 números

- Registar novos números (ignora a partir de quantos_maximo números já estarem registados):
  bingo2 << 10 << 4 << 90 << 4 << 44 << 7 << 22 << 8 << 1 ; //números: 90 e o segundo 4, são ignorados

- Apagar todos os números acima de um dado valor:
  bingo2 >> 20; //Todos números superiores a 20 que já tenham sido sorteados são apagados (saem 44 e 22)

- Ler da consola os números do sorteio (respeitando as regras - não deve ser repetido, estar no intervalo e ter lugar):
  cin >> bingo2; //bingo2 -> vai ficar com o registo dos números que forem introduzidos na consola

- Diminuir a quantidade máxima de números a sortear:
  (bingo2--); // Diminuir a quantidade em 2 unidades, caso o conjunto já tenha excedido esta quantidade, os

(bingo2--)-- ; // Diminuir a quantidade em 2 unidades, caso o conjunto já tenha excedido esta quantidade, os
//últimos elementos devem ser eliminados, até o objeto estar consistente

- Alterar/obter o número numa dada posição do conjunto:

  bingo2[0] = 10;

  cout << bingo2[0]; //independentemente do que estava lá antes, agora aparece 10

```
class Bingo {
public:
Bingo(const string n,const string d, int max, int inf, int sup):
nome(n), data(d), quantos_maximo(max),limite_inf(inf), limite_sup(sup){};
//registar novos numeros
Bingo &operator<<(int n) {
for (auto i : sorteados) {
if (n == sorteados[i])
return *this;
}

sorteados.push_back(n);
return *this;
};

private:
const string data, nome;
int quantos_maximo,limite_inf, limite_sup;
vector <int> sorteados;

};
```

Pergunta 2

Respondida

Nota: 5,00

```
301 class FaltaFruta {  /* ...  */ };
302 class FaltaCafe {  /* ...  */ };
303
304 void testar(string s) {
305     try {
306         if (s == "fruta") {
307             throw FaltaFruta();
308         }
309         else if (s == "cafe") {
310             throw FaltaCafe();
311         }
312     }
313     catch (FaltaFruta & e) {
314         cout << "falta fruta; ";
315     }
316     catch (FaltaCafe & e) {
317         cout << "falta cafe; ";
318         throw;
319     }
320     cout << "mesa; ";
321 }
322
323 int main() {
324     try {
325         testar("cafe");
326         testar("fruta");
327         cout << "sobremesa; ";
328     }
329     catch (FaltaFruta & e) {
330         cout << "iogurte; ";
331     }
332     catch (FaltaCafe & e) {
333         cout << "encomendar; ";
```

```
334         }
335         cout << "receber; \n";
336     }
```

r

Pergunta 3

Respondida

Nota: 5,00

Qual será a saída resultante da execução deste programa?

```
44
45  class SerVivo {
46  public:
47      void alimentar() { cout << "SerVivo alimentar \n"; }
48      virtual void crescer() { cout << "SerVivo crescer \n"; }
49      virtual ~SerVivo() {}
50  };
51  class Elefante : public SerVivo {
52  public:
53      void  alimentar(){ cout << "Elefante alimentar \n"; }
54      void crescer()override { cout << "Elefante crescer \n"; }
55      ~Elefante()override { cout << "Destr Elefante \n"; }
56  };
57
58  int main() {
59      cout << "------\n";
60      SerVivo * x = new Elefante;
61      x->alimentar();
62      x->crescer();
63
64      cout << "------\n";
65      shared_ptr<SerVivo> c = make_shared<Elefante>();
66      c->crescer();
67  }
```

SerVivo alimentar

Elefante crescer

Elefante crescer

Destr Elefante

« PREVIOUS ACTIVITY
Defesas Meta 2 - João Durães

NEXT ACTIVITY
Exam 2nd call - Part 1 of 3 »