10ª aula prática – Tabelas de dispersão

Faça download do ficheiro ***aeda1819_fp10.zip*** da página da disciplina e descomprima-o (contém os ficheiros ***Aposta.h, Jogador.h, Aposta.cpp, Jogador.cpp, Test.cpp***)

- ***Deverá realizar esta ficha respeitando a ordem das alíneas.*** Poderá executar o projeto como CUTE Test quando quiser saber se a implementação que fez é suficiente para passar no teste correspondente.

Enunciado

1. Pretende-se implementar um programa que gera várias apostas de totoloto e compara essas apostas com os números sorteados. Para o efeito, considere a classe **Aposta** que guarda os números de uma aposta numa tabela de dispersão (**unordered_set**). Uma aposta pode ter entre 6 a 12 números.

```
typedef unordered_set<unsigned> tabHInt;

class Aposta
{
    tabHInt numeros;
public:
    Aposta() {};
    void geraAposta(const vector<unsigned> &valores, unsigned n=6);
    bool contem(unsigned num) const;
    unsigned calculaCertos(const tabHInt &sorteio) const;
    tabHInt getNumeros() const { return numeros; }
};
```

a) Implemente o membro-função:

```
void Aposta::geraAposta(const vector<unsigned> &valores, unsigned n)
```

Considere *valores* um vetor de m (m>n) valores gerados aleatoriamente. Esta função cria uma aposta que inclui os primeiros *n* números não repetidos do vetor *valores* e guarda esses números na tabela.

b) Implemente o membro-função:

```
bool Aposta::contem(unsigned num) const
```

Esta função determina se um dado número *num* está contido na aposta.

c) Implemente o membro-função:

```
unsigned Aposta::calculaCertos(const tabHInt &sorteio) const
```

Esta função retorna a quantidade de números certos na aposta, face a um dado *sorteio*.

2) Na continuação do exercício anterior, considere agora a classe **Jogador** que guarda as várias apostas de um jogador.

```
typedef unordered_set<Aposta, apostaHash, apostaHash> tabHAposta;
class Jogador
{
    tabHAposta apostas;
    string nome;
public:
    Jogador(string nm = "anonimo") { nome=nm; }
    void adicionaAposta(const Aposta &ap);
    unsigned apostasNoNumero(unsigned num) const;
    tabHAposta apostasSorteadas(const tabHInt & sorteio) const;
   unsigned getNumApostas() const { return apostas.size(); }
};
```

a) Implemente o membro-função:

```
void Jogador::adicionaAposta(const Aposta &ap)
```

Esta função acrescenta uma dada aposta *ap* ao conjunto de apostas do jogador.

b) Implemente o membro-função:

```
unsigned Jogador::apostasNoNumero(unsigned num) const
```

Esta função determina quantas vezes o jogador apostou em determinado número *num*, no total de apostas efetuadas.

c) Implemente o membro-função:

```
tabHAposta Jogador::apostasPremiadas(const tabHInt &sorteio) const
```

Esta função retorna uma tabela de dispersão contendo as apostas do jogador que são premiadas (isto é, aquelas que possuem mais de 3 valores iguais aos valores de sorteio.