9ª aula prática – Tabelas de dispersão

Faça download do ficheiro ***aeda2021_p09.zip*** da página da disciplina e descomprima-o o (contém a pasta ***lib***, a pasta ***Tests*** com os ficheiros ***bet.h, bet.cpp***, ***player.h, player.cpp*** e ***tests.cpp***, e os ficheiros ***CMakeLists*** e ***main.cpp***) Deverá realizar a ficha respeitando a ordem das alíneas.

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 **Bet** 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 Bet
{
    tabHInt numbers;
public:
    Bet() {};
    void generateBet(const vector<unsigned> &values, unsigned n=6);
    bool contains(unsigned num) const;
    unsigned countRights(const tabHInt& draw) const;
    tabHInt getNumbers() const { return numbers; }
};
```

a) Implemente o membro-função:

```
void Bet::generateBet(const vector<unsigned> &values, unsigned n)
```

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

b) Implemente o membro-função:

```
bool Bet:contains(unsigned num) const
```

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

c) Implemente o membro-função:

```
unsigned Bet::countRights(const tabHInt& draw) const
```

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

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

```
typedef unordered_set<Bet, betHash, betHash> tabHBet;
class Player
{
    tabHBet bets;
    string name;
public:
    Player(string nm = "anonymous") { name=nm; }
    void addBet(const Bet &b);
    unsigned betsInNumber(unsigned num) const;
    tabHBet drawnBets(const tabHInt& draw) const;
   unsigned getNumBets() const { return bets.size(); }
};
```

a) Implemente o membro-função:

```
void Player::addBest(const Best& b)
```

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

b) Implemente o membro-função:

```
unsigned Player::betsInNumber(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:

```
tabHBet Player::drawnBets(const tabHInt& draw) 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.