5ªaula prática - Listas

Faça download do ficheiro ***aeda1920_fp05.zip*** da página da disciplina e descomprima-o (contém a pasta ***lib,*** a pasta ***Tests*** com os ficheiros ***crianca.h, criança.cpp***, ***jogo.h, jogo.cpp*** e ***tests.cpp***, e os ficheiros ***CMakeLists*** e ***main.cpp***)

- Deverá realizar a ficha respeitando a ordem das alíneas.

Enunciado

"*Pim Pam Pum cada bola mata um pra galinha e pro peru quem se livra és mesmo tu*". Recorde este jogo de crianças, cujas regras são as seguintes:

- A primeira criança diz a frase, e em cada palavra vai apontando para cada uma das crianças em jogo (começando em si). Ao chegar ao fim da lista de crianças, volta ao início, ou seja, a ela mesma.
- A criança que está a ser apontada, quando é dita a última palavra da frase, livra-se e sai do jogo. A contagem recomeça na próxima criança.
- Perde o jogo a criança que restar.

Use uma lista (pode usar a classe *list* da STL) para implementar este jogo. Os elementos da lista são objetos da classe **Crianca**, definida abaixo (ver ficheiro *crianca.h*) :

```
class Crianca {
    string nome;
    unsigned idade;
public:
    Crianca();
    Crianca(string nm, unsigned id);
    Crianca(const Crianca &c1);
    unsigned getIdade() const;
    string getNome() const;
    string escreve() const;
};
```

A classe **Jogo** também está previamente declarada:

```
class Jogo
{
    list<Crianca> criancas;
public:
    Jogo();
    Jogo(list<Crianca>& lc2);
    static unsigned int numPalavras(string frase);
    void insereCrianca(const Crianca &c1);
    list<Crianca> getCriancasJogo() const;
    string escreve() const;
    Crianca& perdeJogo(string frase);
    list<Crianca>& inverte();
    list<Crianca> divide(unsigned id);
    void setCriancasJogo(const list<Crianca>& l2);
    bool operator==(Jogo& j2);
    list<Crianca> baralha();
};
```

a) Implemente os construtores da classe **Jogo** e ainda os membros-função:

```
void Jogo::insereCrianca(const Crianca &c1)
```

Esta função adiciona a criança *c1* ao jogo.

```
list<Crianca> Jogo::getCriancasJogo() const
```

Esta função retorna a lista de crianças atualmente em jogo.

```
void Jogo::setCriancasJogo(const list<Crianca> &l1)
```

Esta função coloca em jogo a lista de crianças l1.

b) Implemente o membro-função da classe **Jogo**:

```
string Jogo::escreve() const
```

Esta função retorna uma *string* com todas as crianças que estão em jogo num dado momento. A informação sobre cada criança deve estar no formato "nome : idade" (ver membro-dado `escreve()` da classe `Crianca`).

c) Implemente o membro-função da classe **Jogo**:

```
Crianca& Jogo::perdeJogo(string frase)
```

Esta função realiza o jogo enunciado quando a frase utilizada é *frase* e retorna a criança que perde o jogo. Use o membro-função *numPalavras* (já fornecido) que determina o número de palavras existentes numa *frase* indicada em parâmetro:

```
int Jogo::numPalavras(string frase)
```

d) Implemente o membro-função da classe **Jogo**:

```
list<Crianca>& Jogo::inverte()
```

Esta função inverte a ordem das crianças/ jogadores, na lista *criancas*, em relação à ordem original e retorna a referência para a lista de crianças.

e) Implemente o membro-função da classe **Jogo**:

```
list<Crianca> Jogo::divide(unsigned id)
```

Esta função remove do jogo as crianças de idade superior ao valor *id* especificado como parâmetro, e retorna uma nova lista com as crianças que foram removidas.

f) Implemente o operador de igualdade para a classe **Jogo**:

```
bool Jogo::operator==(Jogo& j2)
```

Dois jogos são considerados iguais se possuem nas suas respetivas listas as mesmas crianças, na mesma ordem.

g) Implemente o membro-função da classe **Jogo**:

```
list<Crianca> Jogo::baralha() const
```

Esta função cria uma nova lista onde as crianças do jogo são colocadas em uma posição determinada aleatoriamente (use a função *rand()* para gerar números aleatórios).