Práctica PRO2: Circuito de torneos de tenis

Autor: Ismael El Basli

Generado por Doxygen 1.9.1

Capítulo 1

Circuito de Torneos de Tenis.

El programa principal se encuentra en el módulo *main.cc*. En base a los datos proporcionados por el enunciado, se ha considerado necesaria la existencia de exactamente 5 clases:

- Jugador
- Cjt_jugadores
- Torneo
- Cjt_torneos
- Cjt_categorias

Capítulo 2

Índice de clases

2.1. Lista de clases

Lista de las clases, estructuras, uniones e interfaces con una breve descripción:

Categoria
: Struct que almacena información sobre una categoria . . . . . ??

Cjt_categorias
: Facilita la gestión de un conjunto de categorias. (obj. 'Categoria') . . . . . ??

Cjt_jugadores
: Facilita la gestión de un conjunto de jugadores (obj. 'Jugador') . . . . . ??

Cjt_torneos
: Facilita la gestión de un conjunto de torneos (obj. 'Jugador') . . . . . ??

Jugador
: Representa un jugador del circuito de torneos de tenis . . . . . ??

Torneo
: Almacena la información de un torneo y facilita su gestión . . . . . ??

Capítulo 3

Indice de archivos

3.1. Lista de archivos

Lista de todos los archivos con descripciones breves:

Cjt_categorias.hh
: Especificacion de la clase Cjt_categorias . . . . . ??

Cjt_jugadores.hh
: Especificación de la clase Cjt_jugadores . . . . . ??

Cjt_torneos.hh
: Especificación de la clase Cjt_torneos . . . . . ??

Jugador.hh
: Especificación de la clase Jugador . . . . . ??

main.cc
: Programa principal para la práctica de PRO2: *Circuito de Torneos de Tenis* . . . . . ??

Torneo.hh
: Especificación de la clase Torneo . . . . . ??

Capítulo 4

Documentación de las clases

4.1. Referencia de la Estructura Categoria

Struct que almacena información sobre una categoria.

Atributos públicos

- string nombre
- vector< int > puntuacion_por_nivel

4.1.1. Descripción detallada

Struct que almacena información sobre una categoria.

Como atributos no se incluye el identificador entero entre 1 y C porque éste será el índice del vector que almacene el conjunto de categorias. El vector de enteros de puntuacion_por_nivel contiene la información de los puntos que ganaria un jugador en un torneo de categoria c al llegar a un nivel k, siendo k un índice de este vector.

Definición en la línea 26 del archivo Cjt_categorias.hh.

4.1.2. Documentación de los datos miembro

4.1.2.1. nombre

```
string Categoria::nombre
```

Definición en la línea 27 del archivo Cjt_categorias.hh.

4.1.2.2. puntuacion_por_nivel

```
vector<int> Categoria::puntuacion_por_nivel
```

Definición en la línea 28 del archivo Cjt_categorias.hh.

La documentación para esta estructura fue generada a partir del siguiente fichero:

- Cjt_categorias.hh

4.2. Referencia de la Clase Cjt_categorias

Facilita la gestión de un conjunto de categorias. (obj. 'Categoria')

Métodos públicos

- Cjt_categorias ()

  Creadora por defecto.
- bool existe_categoria (int c) const

  Consultora que permite saber si existe una categoria en el conjunto.
- int puntos_categoria (int c, int k) const

  Consultora que permite saber los puntos que debe un jugador que ha participado en un torneo de categoria c y que ha llegado a un nivel k.
- void listar_categorias () const

  Operación de escritura.
- void lectura_inicial ()

  Método que se encarga de inicializar los datos del conjunto, dados unos datos por el canal standard de entrada.

Atributos privados

- vector< Categoria > categorias

Atributos privados estáticos

- static const int K

4.2.1. Descripción detallada

Facilita la gestión de un conjunto de categorias. (obj. 'Categoria')

La clase tiene el propósito de gestionar un conjunto de categorias. Ésta permite

Definición en la línea 38 del archivo Cjt_categorias.hh.

4.2.2. Documentación del constructor y destructor

4.2.2.1. Cjt_categorias()

```
Cjt_categorias::Cjt_categorias ( )
```

Creadora por defecto.

Precondición

cierto

Postcondición

El resultado es un Cjt_categorias con su vector de Categoria no definido.

4.2.3. Documentación de las funciones miembro

4.2.3.1. existe_categoria()

```
bool Cjt_categorias::existe_categoria (
            int c ) const
```

Consultora que permite saber si existe una categoria en el conjunto.

Precondición

c es un entero

Postcondición

Se devuelve true si existe una categoria con identificador *c* en el conjunto, false en caso contrario.

4.2.3.2. puntos_categoria()

```
int Cjt_categorias::puntos_categoria (
            int c,
            int k ) const
```

Consultora que permite saber los puntos que debe un jugador que ha participado en un torneo de categoria *c* y que ha llegado a un nivel *k*.

Precondición

c es un identificador de una categoria existente en el conjunto y *k* es un nivel >= 1 y <= K.

Postcondición

se devuelve un entero que representa los puntos que debe ganar un jugador que ha llegado a un nivel *k* en un torneo de categoria *c*.

4.2.3.3. listar_categorias()

```
void Cjt_categorias::listar_categorias ( ) const
```

Operación de escritura.

Precondición

cierto

Postcondición

Se ha listado en el canal standard de salida, por orden creciente de identificador, el nombre y la tabla de puntos por niveles (en orden creciente de nivel) de cada categoria del conjunto.

4.2.3.4. lectura_inicial()

```
void Cjt_categorias::lectura_inicial ( )
```

Método que se encarga de inicializar los datos del conjunto, dados unos datos por el canal standard de entrada.

Precondición

Está disponible en el canal standard de entrada un número de categorias C >= 0 seguido de un número máximo de niveles K >= 4. Después hay una secuencia de C strings con los nombres asociados a las categorías identificadas por los valores entre 1 y C en orden creciente (de identificador, no de nombre). Luego hay disponibles C × K enteros mayores o iguales que 0, que serán los puntos por categoría y nivel, ordenados crecientemente por categorías y dentro de cada categoría ordenados crecientemente por nivel.

Postcondición

El parámetro implícito ha sido modificado de forma que se han almacenado todas las categorias con su correspondiente información.

4.2.4. Documentación de los datos miembro

4.2.4.1. categorias

```
vector<Categoria> Cjt_categorias::categorias  [private]
```

Definición en la línea 41 del archivo Cjt_categorias.hh.

4.2.4.2. K

```
const int Cjt_categorias::K  [static], [private]
```

Definición en la línea 42 del archivo Cjt_categorias.hh.

La documentación para esta clase fue generada a partir del siguiente fichero:

- Cjt_categorias.hh

4.3. Referencia de la Clase Cjt_jugadores

Facilita la gestión de un conjunto de jugadores (obj. 'Jugador')

Métodos públicos

- Cjt_jugadores ()

  Creadora por defecto.
- void nuevo_jugador (const string &p)

  Modificadora que añade un nuevo jugador al conjunto de jugadores.
- void baja_jugador (const string &p)

  Modificadora que da de baja un jugador del conjunto de jugadores.
- void restar_puntos (const vector< Jugador > &datos)

  Modificadora que se encarga de restar puntos a ciertos jugadores dado un vector con es información, además de mantener el orden definido para el ranking.
- void actualizar_ranking (const vector< Jugador > &datos)

  Modificadora que se encarga de actualizar las estadísticas de ciertos jugadores dado un vector con esa información, además de mantener el orden definido para el ranking.
- bool existe_jugador (const string &p) const

  Consultora que permite saber si un jugador ya existe en el conjunto.
- void listar_ranking () const

  Operación de escritura.
- void listar_jugadores () const

  Operación de escritura.
- void consultar_jugador (const string &p) const

  Operación de escritura.
- void lectura_inicial ()

  Método que se encarga de inicializar los atributos del conjunto, dados unos datos por el canal standard de entrada.

Atributos privados

- int P
- map< string, Jugador > jugadores
- vector< map< string, Jugador >::iterator > ranking

4.3.1. Descripción detallada

Facilita la gestión de un conjunto de jugadores (obj. 'Jugador')

La clase tiene el propósito de gestionar un conjunto de jugadores. Ésta permite añadir y dar de baja jugadores, así como listarlos, ya sea en conjunto o de forma individual. Además almacena en un ranking a los jugadores en función de sus puntos.

Definición en la línea 25 del archivo Cjt_jugadores.hh.

4.3.2. Documentación del constructor y destructor

4.3.2.1. Cjt_jugadores()

```
Cjt_jugadores::Cjt_jugadores ( )
```

Creadora por defecto.

Precondición

cierto

Postcondición

El resultado es un Cjt_jugadores con el numero total de jugadores a 0 y las estructuras que se encargan de almacenar los datos de los jugadores se encuentran vacías, sin elementos.

4.3.3. Documentación de las funciones miembro

4.3.3.1. nuevo_jugador()

```
void Cjt_jugadores::nuevo_jugador (
            const string & p )
```

Modificadora que añade un nuevo jugador al conjunto de jugadores.

Precondición

p es un identificador válido de un jugador no existente en el conjunto de jugadores.

Postcondición

El número total de jugadores se ha incrementado en uno, y el jugador con identificador *p* se ha añadido al conjunto, con estadísticas a cero y última posición en el ranking. Posteriormente a esto, se imprime por el canal standard de salida el número total de jugadores.

4.3.3.2. baja_jugador()

```
void Cjt_jugadores::baja_jugador (
            const string & p )
```

Modificadora que da de baja un jugador del conjunto de jugadores.

Precondición

p es un identificador válido de un jugador que se encuentra en el conjunto de jugadores.

Postcondición

El ranking ha sido actualizado desplazando una posición hacia arriba a los jugadores siguientes del ranking, el número total de jugadores ha decrementado en uno, y el jugador con identificador *p* ha sido eliminado del conjunto.

4.3.3.3. restar_puntos()

```
void Cjt_jugadores::restar_puntos (
            const vector< Jugador > & datos )
```

Modificadora que se encarga de restar puntos a ciertos jugadores dado un vector con es información, además de mantener el orden definido para el ranking.

Precondición

datos es un vector de Jugadores que tiene la información de todas sus estadísticas en un torneo concreto.

Postcondición

Se han restado a los jugadores respectivos del conjunto de jugadores los puntos que habían ganado en el torneo en cuestión (información disponible en el vector), además de que se mantiene el orden definido para el ranking.

4.3.3.4. actualizar_ranking()

```
void Cjt_jugadores::actualizar_ranking (
            const vector< Jugador > & datos )
```

Modificadora que se encarga de actualizar las estadísticas de ciertos jugadores dado un vector con esa información, además de mantener el orden definido para el ranking.

Precondición

datos es un vector de Jugadores que tiene la información de todas sus estadísticas en un torneo concreto.

Postcondición

Se han actualizado las estadísticas de los jugadores respectivos del conjunto de jugadores, además de que se mantiene el orden definido para el ranking.

4.3.3.5. existe_jugador()

```
bool Cjt_jugadores::existe_jugador (
            const string & p ) const
```

Consultora que permite saber si un jugador ya existe en el conjunto.

Precondición

p es un identificador válido de un jugador.

Postcondición

Se devuelve true en caso de que el jugador con identificador *p* exista en el conjunto, false en caso contrario.

4.3.3.6. listar_ranking()

```
void Cjt_jugadores::listar_ranking ( ) const
```

Operación de escritura.

Precondición

cierto

Postcondición

Se ha listado en el canal standard de salida, por orden creciente del ranking vigente, la posicion, el nombre y los puntos de cada jugador del conjunto.

4.3.3.7. listar_jugadores()

```
void Cjt_jugadores::listar_jugadores ( ) const
```

Operación de escritura.

Precondición

cierto

Postcondición

Se ha listado en el canal standard de salida, por orden creciente de identificador (nombre), la posición en el ranking, los puntos y el resto de estadísticas de cada jugador del conjunto.

4.3.3.8. consultar_jugador()

```
void Cjt_jugadores::consultar_jugador (
            const string & p ) const
```

Operación de escritura.

Precondición

p es un identificador válido de un jugador existente en el conjunto.

Postcondición

Se ha listado en el canal standard de salida el nombre, la posición en el ranking, los puntos y el resto de estadísticas del jugador con identificador *p*.

4.3.3.9. lectura_inicial()

```
void Cjt_jugadores::lectura_inicial ( )
```

Método que se encarga de inicializar los atributos del conjunto, dados unos datos por el canal standard de entrada.

Precondición

Está disponible en el canal standard de entrada un número P >= 0, seguido de una secuencia de P strings con los nombres que identifican a los jugadores (sin repeticiones).

Postcondición

El parámetro implícito ha sido modificado de forma que se han almacenado P jugadores en las estructuras que se encargan de facilitar su gestión. Todas las estadísticas de los jugadores, a excepción de su posición en el ranking, son inicializadas a cero.

4.3.4. Documentación de los datos miembro

4.3.4.1. P

```
int Cjt_jugadores::P  [private]
```

Definición en la línea 28 del archivo Cjt_jugadores.hh.

4.3.4.2. jugadores

```
map<string,Jugador> Cjt_jugadores::jugadores  [private]
```

Definición en la línea 29 del archivo Cjt_jugadores.hh.

4.3.4.3. ranking

```
vector<map<string,Jugador>::iterator> Cjt_jugadores::ranking  [private]
```

Definición en la línea 30 del archivo Cjt_jugadores.hh.

La documentación para esta clase fue generada a partir del siguiente fichero:

- Cjt_jugadores.hh

4.4. Referencia de la Clase Cjt_torneos

Facilita la gestión de un conjunto de torneos (obj. 'Jugador')

Métodos públicos

- Cjt_torneos ()
  Creadora por defecto.
- void nuevo_torneo (const string &t, int c)
  Modificadora que añade un nuevo torneo al conjunto de torneos.
- void baja_torneo (const string &t, Cjt_jugadores &j)
  Modificadora que da de baja a un torneo del conjunto de torneos.
- void iniciar_torneo (const string &t)
  Modificadora que se encarga de "iniciar" un torneo.
- void finalizar_torneo (const string &t, Cjt_jugadores &j)
  Modificadora que se encarga de "finalizar" un torneo.
- bool existe_torneo (const string &t)
  Consultora que permite saber si un torneo ya existe en el conjunto.
- void listar_torneos () const
  Operación de escritura.
- void lectura_inicial ()
  Método que se encarga de inicializar los atributos del conjunto, dados unos datos por el canal standard de entrada.

Atributos privados

- int T
- map< string, Torneo > torneos

4.4.1. Descripción detallada

Facilita la gestión de un conjunto de torneos (obj. 'Jugador')

La clase tiene el propósito de gestionar un conjunto de torneos. Ésta permite añadir, dar de baja, iniciar y finalizar torneos, así como listarlos por orden creciente de identificador (nombre).

Definición en la línea 25 del archivo Cjt_torneos.hh.

4.4.2. Documentación del constructor y destructor

4.4.2.1. Cjt_torneos()

```
Cjt_torneos::Cjt_torneos ( )
```

Creadora por defecto.

Precondición

cierto

Postcondición

El resultado es un Cjt_torneos con el número total de torneos a 0 y la estructura que se encarga de almacenar los torneos se encuentra vacía, sin elementos.

4.4.3. Documentación de las funciones miembro

4.4.3.1. nuevo_torneo()

```
void Cjt_torneos::nuevo_torneo (
            const string & t,
            int c )
```

Modificadora que añade un nuevo torneo al conjunto de torneos.

Precondición

t es un identificador válido de un torneo no existente en el conjunto de torneos, y *c* es un entero entre 1 y C que hace referencia a una categoria válida de torneo.

Postcondición

El número total de torneos se ha incrementado en uno, y el torneo con identificador *t* de categoria *c* se ha añadido al conjunto. Posteriormente a esto, se imprime por el canal standard de salida el número total de torneos.

4.4.3.2. baja_torneo()

```
void Cjt_torneos::baja_torneo (
            const string & t,
            Cjt_jugadores & j )
```

Modificadora que da de baja a un torneo del conjunto de torneos.

Precondición

t es un identificador válido de un torneo que se encuentra en el conjunto de torneos.

Postcondición

El ranking ha sido actualizado después de restar los puntos conseguidos por los jugadores de una posible edición pasada del torneo con identificador *t*, el número total de torneos ha decrementado en uno, y el torneo en cuestión ha sido eliminado del conjunto. Posteriormente a esto, se imprime por el canal standard de salida el número total de torneos.

4.4.3.3. iniciar_torneo()

```
void Cjt_torneos::iniciar_torneo (
            const string & t )
```

Modificadora que se encarga de "iniciar" un torneo.

Precondición

cierto, ya que se garantiza que el torneo *t* existe en el circuito.

Postcondición

Se ha leído la inscripción del torneo *t* y se ha confeccionado e impreso el cuadro de emparejamientos de los jugadores inscritos.

4.4.3.4. finalizar_torneo()

```
void Cjt_torneos::finalizar_torneo (
            const string & t,
            Cjt_jugadores & j )
```

Modificadora que se encarga de "finalizar" un torneo.

Precondición

cierto, ya que se garantiza que el torneo *t* existe en el circuito y que previamente se ha ejecutado el comando *iniciar_torneo*.

Postcondición

Se han leído los resultados del torneo *t*, se ha producido e impreso el cuadro oficial de resultados y se han listado los puntos para el ránking ganados por cada uno de los participantes, por orden de ranking en el momento de iniciar el torneo. Además, se han actualizado el ranking y las estadísticas de los jugdores.

4.4.3.5. existe_torneo()

```
bool Cjt_torneos::existe_torneo (
            const string & t )
```

Consultora que permite saber si un torneo ya existe en el conjunto.

Precondición

t es un identificador válido de un torneo

Postcondición

Se devuelve true en caso de que el jugador con identificador *t* exista en el conjunto, false en caso contrario.

4.4.3.6. listar_torneos()

```
void Cjt_torneos::listar_torneos ( ) const
```

Operación de escritura.

Precondición

cierto

Postcondición

Se ha listado en el canal standard de salida, por orden creciente de identificador (nombre), el nombre y la categoria de cada torneo del circuito.

4.4.3.7. lectura_inicial()

```
void Cjt_torneos::lectura_inicial ( )
```

Método que se encarga de inicializar los atributos del conjunto, dados unos datos por el canal standard de entrada.

Precondición

Está disponible en el canal standard de entrada un número T $>=$ 0, seguido de una secuencia de T pares de string t y entero c, donde t será el nombre que identifica al torneo (sin repeticiones) y c la categoria entre 1 y C a la que pertenece el torneo.

Postcondición

El parámetro implícito ha sido modificado de forma que se han almacenado T torneos en la estructura que se encarga de faciltar su gestión.

4.4.4. Documentación de los datos miembro

4.4.4.1. T

```
int Cjt_torneos::T  [private]
```

Definición en la línea 28 del archivo Cjt_torneos.hh.

4.4.4.2. torneos

```
map<string, Torneo> Cjt_torneos::torneos  [private]
```

Definición en la línea 29 del archivo Cjt_torneos.hh.

La documentación para esta clase fue generada a partir del siguiente fichero:

- Cjt_torneos.hh

4.5. Referencia de la Clase Jugador

Representa un jugador del circuito de torneos de tenis.

Métodos públicos

- Jugador ()
  Creadora por defecto.
- void modificar_partidos (int g, int p)
  Modificadora del par que almacena la información de los partidos ganados/perdidos.
- void modificar_sets (int g, int p)
  Modificadora del par que almacena la información de los sets ganados/perdidos.
- void afegir_td ()
  Modificadora del número de torneos disputados.
- void modificar_puntos (int p)
  Modificadora de los puntos del jugador.
- void modificar_posicion (int p)
  Modificadora de la posicion del jugador.
- int consultar_td () const
  Consultora del número de torneos disputados.
- int consultar_pg () const
  Consultora del número de partidos ganados.
- int consultar_pp () const
  Consultora del número de partidos perdidos.
- int consultar_sg () const
  Consultora del número de sets ganados.
- int consultar_sp () const
  Consultora del número de sets perdidos.
- int consultar_pos () const
  Consultora de la posicion.
- int consultar_puntos () const
  Consultora de los puntos.
- void escribir () const
  Operación de escritura.

Atributos privados

- int torneos_disputados
- int posicion
- int puntos
- Estadisticas partidos
- Estadisticas sets

4.5.1. Descripción detallada

Representa un jugador del circuito de torneos de tenis.

Dispone de ciertos atributos que definen al jugador. Como detalle, todas las operaciones son de **coste constante**.

Definición en la línea 22 del archivo Jugador.hh.

4.5.2. Documentación del constructor y destructor

4.5.2.1. Jugador()

```
Jugador::Jugador ( )
```

Creadora por defecto.

Se ejecuta automáticamente al declarar un jugador.

Precondición

cierto

Postcondición

El resultado es un jugador el cual todos sus atributos son cero.

4.5.3. Documentación de las funciones miembro

4.5.3.1. modificar_partidos()

```
void Jugador::modificar_partidos (
            int g,
            int p )
```

Modificadora del par que almacena la información de los partidos ganados/perdidos.

Precondición

g es un entero que representa los partidos ganados y *p* los perdidos.

Postcondición

El parámetro implícito pasa a tener *g* partidos ganados y *p* partidos perdidos.

4.5.3.2. modificar_sets()

```
void Jugador::modificar_sets (
            int g,
            int p )
```

Modificadora del par que almacena la información de los sets ganados/perdidos.

Precondición

g es un entero que representa los sets ganados y *p* los perdidos.

Postcondición

El parámetro implícito pasa a tener *g* sets ganados y *p* sets perdidos.

4.5.3.3. afegir_td()

```
void Jugador::afegir_td ( )
```

Modificadora del número de torneos disputados.

Precondición

cierto

Postcondición

El número de torneos disputados del parámetro implícito ha aumentado en una unidad.

4.5.3.4. modificar_puntos()

```
void Jugador::modificar_puntos (
            int p )
```

Modificadora de los puntos del jugador.

Precondición

p es un número >= 0

Postcondición

El parámetro implícito pasa a tener *p* puntos.

4.5.3.5. modificar_posicion()

```
void Jugador::modificar_posicion (
            int p )
```

Modificadora de la posicion del jugador.

Precondición

p es un número >= 1 y <= num. jugadores del circuito.

Postcondición

El parámetro implícito pasa a tener posición *p*.

4.5.3.6. consultar_td()

```
int Jugador::consultar_td ( ) const
```

Consultora del número de torneos disputados.

Precondición

cierto

Postcondición

El resultado es el número de torneos disputados del parámetro implícito.

4.5.3.7. consultar_pg()

```
int Jugador::consultar_pg ( ) const
```

Consultora del número de partidos ganados.

Precondición

cierto

Postcondición

El resultado es el número de partidos ganados del parámetro implícito.

4.5.3.8. consultar_pp()

```
int Jugador::consultar_pp ( ) const
```

Consultora del número de partidos perdidos.

Precondición

cierto

Postcondición

El resultado es el número de partidos perdidos del parámetro implícito.

4.5.3.9. consultar_sg()

```
int Jugador::consultar_sg ( ) const
```

Consultora del número de sets ganados.

Precondición

cierto

Postcondición

El resultado es el número de sets ganados del parámetro implícito.

4.5.3.10. consultar_sp()

```
int Jugador::consultar_sp ( ) const
```

Consultora del número de sets perdidos.

Precondición

cierto

Postcondición

El resultado es el número de sets perdidos del parámetro implícito.

4.5.3.11. consultar_pos()

```
int Jugador::consultar_pos ( ) const
```

Consultora de la posicion.

Precondición

cierto

Postcondición

El resultado es el número que hace referencia a la posición del parámetro implícito dentro del ranking.

4.5.3.12. consultar_puntos()

```
int Jugador::consultar_puntos ( ) const
```

Consultora de los puntos.

Precondición

cierto

Postcondición

El resultado es el número que hace referencia a los puntos del parámetro implícito.

4.5.3.13. escribir()

```
void Jugador::escribir ( ) const
```

Operación de escritura.

Precondición

cierto

Postcondición

Se han escrito los atributos del parámetro implícito en el canal standard de salida.

4.5.4. Documentación de los datos miembro

4.5.4.1. torneos_disputados

```
int Jugador::torneos_disputados  [private]
```

Definición en la línea 27 del archivo Jugador.hh.

4.5.4.2. posicion

```
int Jugador::posicion  [private]
```

Definición en la línea 28 del archivo Jugador.hh.

4.5.4.3. puntos

```
int Jugador::puntos  [private]
```

Definición en la línea 29 del archivo Jugador.hh.

4.5.4.4. partidos

```
Estadisticas Jugador::partidos  [private]
```

Definición en la línea 30 del archivo Jugador.hh.

4.5.4.5. sets

```
Estadisticas Jugador::sets  [private]
```

Definición en la línea 31 del archivo Jugador.hh.

La documentación para esta clase fue generada a partir del siguiente fichero:

- Jugador.hh

4.6. Referencia de la Clase Torneo

Almacena la información de un torneo y facilita su gestión.

Métodos públicos

- Torneo ()

  Creadora por defecto.
- Torneo (int c)

  Creadora con valores concretos.
- void generar_enfr ()

  Modificadora del cuadro de emparejamientos.
- vector< Jugador > info_participantes () const

  Consultora que devuelve la estructura que almacena la información de los participantes del torneo. (tiene el objetivo de poder acceder a ésta información desde otra clase, para poder actualizar de forma adecuada el ranking y las estadísticas de los jugadors, ya que luego ésta información se pierde al leer los nuevos participantes de la posible nueva edición).
- void imprimir_enfrentamientos () const

  Operación de escritura.
- void imprimir_resultados () const

  Operación de escritura.
- void listar_puntos () const

  Operación de escritura.
- void leer_inscritos ()

  Lectura de los jugadores inscritos en el torneo.
- void leer_resultados ()

  Lectura de los resultados del torneo.

Atributos privados

- int categoria
- vector< Jugador > inscritos
- BinTree< string > enfrentamientos
- BinTree< string > resultados

4.6.1. Descripción detallada

Almacena la información de un torneo y facilita su gestión.

Gestiona la información de un torneo:

- Identificador de su categoria
- Jugadores inscritos
- Cuadro de emparejamientos
- Cuadro de resultados
- Puntos ganados por los jugadores (y estadísticas "locales")

Definición en la línea 30 del archivo Torneo.hh.

4.6.2. Documentación del constructor y destructor

4.6.2.1. Torneo() [1/2]

```
Torneo::Torneo ( )
```

Creadora por defecto.

Precondición

cierto

Postcondición

El resultado es un torneo sin categoria definida, y el resto de atributos son vacíos.

4.6.2.2. Torneo() [2/2]

```
Torneo::Torneo (
            int c )
```

Creadora con valores concretos.

Precondición

c es una categoria válida entre 1 y C

Postcondición

El resultado es un torneo de categoria *c*. El resto de atributos son vacíos.

4.6.3. Documentación de las funciones miembro

4.6.3.1. generar_enfr()

```
void Torneo::generar_enfr ( )
```

Modificadora del cuadro de emparejamientos.

Precondición

Ya se han leído los participantes del torneo.

Postcondición

La estructura encargada de almacenar la información de los emparejamientos ha sido inicializada en base a la información de los jugadores.

4.6.3.2. info_participantes()

```
vector<Jugador> Torneo::info_participantes ( ) const
```

Consultora que devuelve la estructura que almacena la información de los participantes del torneo. (tiene el objetivo de poder acceder a ésta información desde otra clase, para poder actualizar de forma adecuada el ranking y las estadísticas de los jugadors, ya que luego ésta información se pierde al leer los nuevos participantes de la posible nueva edición).

Precondición

cierto

Postcondición

Se devuelve la estructura que almacena toda la información de los jugadores del torneo.

4.6.3.3. imprimir_enfrentamientos()

```
void Torneo::imprimir_enfrentamientos ( ) const
```

Operación de escritura.

Precondición

cierto

Postcondición

Se ha imprimido por el canal standard de salida el cuadro de emparejamientos de los jugadores inscritos.

4.6.3.4. imprimir_resultados()

```
void Torneo::imprimir_resultados ( ) const
```

Operación de escritura.

Precondición

Se han leído previamente los resultados.

Postcondición

Se ha imprimido por el canal standard de salida el cuadro de resultados, una vez finalizado el torneo.

4.6.3.5. listar_puntos()

```
void Torneo::listar_puntos ( ) const
```

Operación de escritura.

Precondición

cierto

Postcondición

Se han listado por el canal standard de salida los puntos para el ranking ganados por cada uno de los participantes, por orden del ranking de los jugadores en el momento de iniciar el torneo.

4.6.3.6. leer_inscritos()

```
void Torneo::leer_inscritos ( )
```

Lectura de los jugadores inscritos en el torneo.

Precondición

Esta disponible en el canal standard de entrada la información de los jugadores inscritos, concretamente un número n de jugadores inscritos, seguido de n enteros con sus posiciones en el ranking vigente, ordenadas crecientemente.

Postcondición

El parámetro implícito ha sido modificado de forma que la estructura que se encarga de almacenar los jugadores inscritos ha sido inicializada con los jugadores leídos.

4.6.3.7. leer_resultados()

```
void Torneo::leer_resultados ( )
```

Lectura de los resultados del torneo.

Precondición

Está disponible en el canal standard de entrada la información de los resultados del torneo, en forma de árbol binario de strings, concretamente en preorden.

Postcondición

El parámetro implícito ha sido modificado de forma que la estructura que se encarga de almacenar los puntos ganados por los jugadores ha sido actualizada en base a los datos leídos. Además, también se ha modificado el BinTree que almacena la información de los resultados.

4.6.4. Documentación de los datos miembro

4.6.4.1. categoria

```
int Torneo::categoria  [private]
```

Definición en la línea 35 del archivo Torneo.hh.

4.6.4.2. inscritos

```
vector<Jugador> Torneo::inscritos  [private]
```

Definición en la línea 36 del archivo Torneo.hh.

4.6.4.3. enfrentamientos

```
BinTree<string> Torneo::enfrentamientos  [private]
```

Definición en la línea 37 del archivo Torneo.hh.

4.6.4.4. resultados

```
BinTree<string> Torneo::resultados  [private]
```

Definición en la línea 38 del archivo Torneo.hh.

La documentación para esta clase fue generada a partir del siguiente fichero:

- Torneo.hh

Capítulo 5

Documentación de archivos

5.1. Referencia del Archivo Cjt_categorias.hh

Especificacion de la clase Cjt_categorias.

Clases

- struct Categoria

  Struct que almacena información sobre una categoria.
- class Cjt_categorias

  Facilita la gestión de un conjunto de categorias. (obj. 'Categoria')

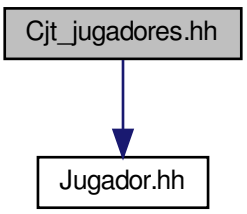
5.1.1. Descripción detallada

Especificacion de la clase Cjt_categorias.

5.2. Referencia del Archivo Cjt_jugadores.hh

Especificación de la clase Cjt_jugadores.

Dependencia gráfica adjunta para Cjt_jugadores.hh:



Clases

- class Cjt_jugadores

  Facilita la gestión de un conjunto de jugadores (obj. 'Jugador')

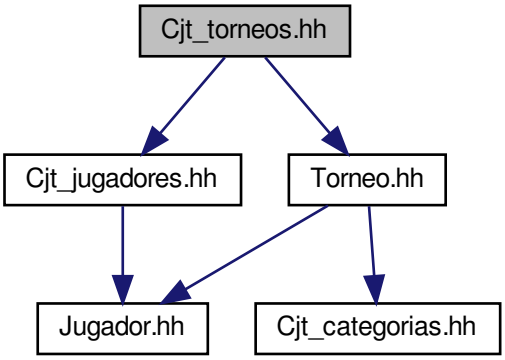
5.2.1. Descripción detallada

Especificación de la clase Cjt_jugadores.

5.3. Referencia del Archivo Cjt_torneos.hh

Especificación de la clase Cjt_torneos.

Dependencia gráfica adjunta para Cjt_torneos.hh:



Clases

- class Cjt_torneos

  Facilita la gestión de un conjunto de torneos (obj. 'Jugador')

5.3.1. Descripción detallada

Especificación de la clase Cjt_torneos.

5.4. Referencia del Archivo Jugador.hh

Especificación de la clase Jugador.

Clases

- class Jugador

  Representa un jugador del circuito de torneos de tenis.

typedefs

- typedef pair< int, int > Estadisticas

  Pair que identifica las estadisticas de partidos y sets. (ganados-perdidos)

5.4.1. Descripción detallada

Especificación de la clase Jugador.

5.4.2. Documentación de los 'typedefs'

5.4.2.1. Estadisticas

```
Estadisticas
```

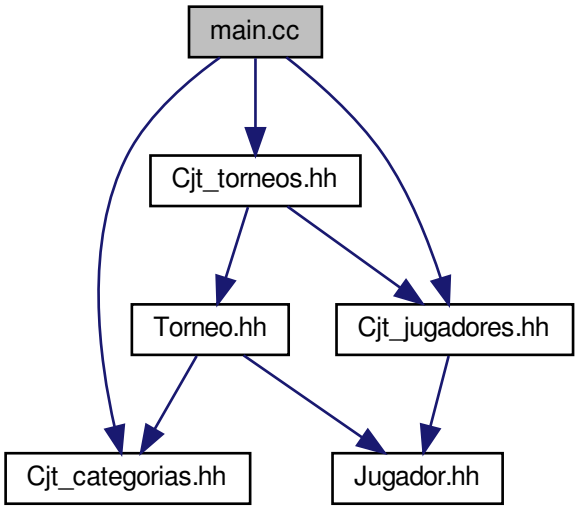
Pair que identifica las estadisticas de partidos y sets. (ganados-perdidos)

Definición en la línea 13 del archivo Jugador.hh.

5.5. Referencia del Archivo main.cc

Programa principal para la práctica de PRO2: *Circuito de Torneos de Tenis*.

Dependencia gráfica adjunta para main.cc:



Funciones

- int main ()

  Programa principal para la práctica de PRO2: Circuito de Torneos de Tenis.

5.5.1. Descripción detallada

Programa principal para la práctica de PRO2: *Circuito de Torneos de Tenis*.

Como bien da a entender el enunciado, se supone que los datos leídos siempre son correctos, de modo que no se incluyen comprobaciones al respecto.

5.5.2. Documentación de las funciones

5.5.2.1. main()

```
int main ( )
```

Programa principal para la práctica de PRO2: *Circuito de Torneos de Tenis*.

Definición en la línea 34 del archivo main.cc.

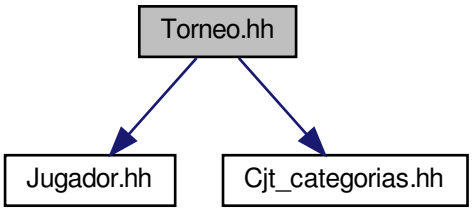
```
35 {
36   Cjt_categorias c;
37   Cjt_torneos t;
38   Cjt_jugadores j;
39
40   c.lectura_inicial();
41   t.lectura_inicial();
42   j.lectura_inicial();
43
44   string cm;
45   cin  cm;

46 while (cm != "fin") {

47 string p;

48 if (cm == "nuevo_jugador" or cm == "nj") {

49 cin  p;

50 if (j.existe_jugador(p)) cout « "error" “ endl;

»”51 else j.nuevo_jugador(p);

»”52 }

»”53 else if (cm == "nuevo_torneo" or cm == "nt") {

»”54 int cat;

»”55 cin ” p » cat;

56 if (not c.existe_categoria(cat)) cout « "error" “ endl;
```

```
»"57 else if (t.existe_torneo(p)) cout ‘ "error"  endl;
»"58 else t.nuevo_torneo(p, cat);
»"59 }
»"60 else if (cm == "baja_jugador" or cm == "bj") {
»"61 cin  p;
»"62 if (not j.existe_jugador(p)) cout  "error"  endl;
»"63 else j.baja_jugador(p);
»"64 }
»"65 else if (cm == "baja_torneo" or cm == "bt") {
»"66 cin  p;
»"67 if (not t.existe_torneo(p)) cout  "error"  endl;
»"68 else t.baja_torneo(p, j);
»"69 }
»"70 else if (cm == "iniciar_torneo" or cm == "it") {
»"71 cin  p;
»"72 t.iniciar_torneo(p);
»"73 }
»"74 else if (cm == "finalizar_torneo" or cm == "ft") {
»"75 cin  p;
»"76 t.finalizar_torneo(p, j);
»"77 }
»"78 else if (cm == "listar_ranking" or cm == "lr") {
»"79 j.listar_ranking();
»"80 }
»"81 else if (cm == "listar_jugadores" or cm == "lj") {
»"82 j.listar_jugadores();
»"83 }
»"84 else if (cm == "consultar_jugador" or cm == "cj") {
»"85 cin  p;
»"86 if (not j.existe_jugador(p)) cout  "error"  endl;
»"87 else j.consultar_jugador(p);
»"88 }
»"89 else if (cm == "listar_torneos" or cm == "lt") {
»"90 t.listar_torneos();
»"91 }
»"92 else if (cm == "listar_categorias" or cm == "lc") {
»"93 c.listar_categorias();
»"94 }
»"95 cin  cm;
»"96 }
»"97 }
```

»”’5.6. Referencia del Archivo Torneo.hh

»”’Especificación de la clase Torneo.

»”’Dependencia gráfica adjunta para Torneo.hh:



»”’

»”’Clases

- class Torneo

  »”’Almacena la información de un torneo y facilita su gestión.

»”’5.6.1. Descripción detallada

»”’Especificación de la clase Torneo.